DTIC FILE COPY

(2)

Report SOAS-FRPT

AD-A209 854

SUBMARINE OPERATIONAL AUTOMATION SYSTEM PHASE I FINAL TECHNICAL REPORT

M. D. Prince
Lockheed Aeronautical Systems Company
Advanced Systems Development Center
86 South Cobb Drive
Marietta, GA 30063

1989 May 31

Final Technical Report for November 1988 - May 1989
Contract No. MDA972-89-C-0006

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Sponsored by
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
Naval Technology Office
Submarine Technology Program
ARPA Order No. 6661
Issued by DARPA/CMO under Contract MDA972-88-C-0069

89 6 13 148

Report SOAS-FRPT

SUBMARINE OPERATIONAL AUTOMATION SYSTEM PHASE I FINAL TECHNICAL REPORT

M. D. Prince
Lockheed Aeronautical Systems Company
Advanced Systems Development Center
86 South Cobb Drive
Marietta, GA 30063

1989 May 31

Final Technical Report for November 1988 - May 1989
Contract No. MDA972-89-C-0006

Sponsored by
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
Naval Technology Office
Submarine Technology Program
ARPA Order No. 6661
Issued by DARPA/CMO under Contract MDA972-88-C-0069

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SOAS-FRPT	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Lockheed Aeronautical Systems Company; 6b. OFFICE SYMBOL (If applicable) LASC	7a. NAME OF MONITORING ORGANIZATION Defense Advanced Research Projects Agency Submarine Technology Program Support Office
6c. ADDRESS (City, State, and ZIP Code) 86 South Cobb Drive Marietta, GA 30063 ATTN: M. D. Prince, D/73-60, Z/410	7b. ADDRESS (City, State, and ZIP Code) 1515 Wilson Blvd. Suite 705 Arlington, VA 22209
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Advanced Research Projects Agency, Contracts Mgt Office; 8b. OFFICE SYMBOL (If applicable) DARPA/CMO	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA972-89-C-0006
8c. ADDRESS (City, State, and ZIP Code) 1400 Wilson Blvd. Arlington, VA 22209-2308 ATTN: G. E. Mayberry	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Submarine Operational Automation System, Phase I, Final Technical Report

12. PERSONAL AUTHOR(S)
M. D. Prince

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 11/88 TO 5/89	1989 May 31	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	SOAS, Artificial Intelligence, Submarine Technology Knowledge-Based System Expert System

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED [✓ SAME AS RPT.] ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code); 22c. OFFICE SYMBOL

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd.)

TABLE OF CONTENTS (Cont'd.)

LIST OF FIGURES

LIST OF FIGURES (Cont'd.)

1.0 INTRODUCTION AND SUMMARY

1.1 IDENTIFICATION

This report details the approach and progress during Phase I of the Submarine Operational Automation System (SOAS) program, culminating in a system demonstration. A summary of the problem and approach, both to the design of system as a whole and to the development of individual SOAS subsystem functions, will be presented. The results of pursuing these approaches and the strategy for the next phases of SOAS will be discussed.

1.2 PROGRAM OVERVIEW

The first phase of this program comprised the development and demonstration of proposed concepts for the Submarine Operational Automation System. The objective of the working demonstrations was to show the potential effectiveness and technical feasibility of a future full-scale, knowledge-based command and control system for advanced submarines to enhance the tactical and operational performance of the platform.

The concept development system and demonstration were required to incorporate the following specific features:

- The capability to perform or assist is useful and meaningful submarine command-level functions
- The incorporation and demonstration of innovative and state-of-the-art computer systems, software, and displays
- The development, incorporation, and adaptation of innovative and state-of-the-art expert- and/or knowledge-based shells, software, and high-level languages to the submarine command system situation
- Effective man-machine displays and interfaces and the use and incorporation thereof
- The compilation and development of a suitable knowledge base of submarine information
- A system configuration suitable for expansion, revision, and upgrade and interface to ship systems and possibly other expert- or knowledge-based systems.

In addition to the concept development system and demonstration, the scope of work for Phase I included the delivery of a System Development Plan describing the proposed approach to Phases II and III, which together will lead to a command-level operational automation system intended to be installed on a submarine for final testing and demonstration. The System Development Plan, while discussed briefly in this report, is presented in detail under separate cover.

1.3 SUMMARY OF APPROACH

The approach to the development of the Submarine Operational Automation System taken by the Lockheed team is summarized by the following tasks:

- Scenario Selection
- Submarine Systems Definition
- Mission Task Analysis
- SOAS Functions Selection
- Subsystem Specification
- Technology Assessment
- Prototyping and Incremental Integration
- Software Integration Management
- Demonstrations and Evaluations

Scenario Selection - A battle scenario, including representative 1995 targets, threats, Blue and Orange assets, Blue and Red submarine capabilities, tactics, rules of engagement, and mission activities was developed, primarily by tailoring scenarios provided by the Naval Underwater Systems Center (NUSC). These scenarios were tailored to include events that would not only stress the commander, the crew, and the weapons systems, but also provide the opportunity to demonstrate specific commander-aiding functions which SOAS is intended to provide. The scenario events also became the basis for setting the requirements for the simulation environment, which is the test and demonstration environment for the Submarine Operational Automation System. The scenario used for the system demonstration is detailed in the Demonstration Plan document for the Submarine Operational Automation System Program (December, 1989).

Submarine Systems Definition - Specific submarine performance characteristics, system capabilities, and weapons numbers and types were elected as the basis for developing the knowledge bases of the Submarine Operational Automation System.

Mission Task Analysis - An analysis of the attack center personnel and submarine system tasks was performed. This, along with the technology base assessment, became the basis for the selection of candidate Submarine Operational Automation System functions.

SOAS Functions Selection - Using the task list produced from the task analysis, candidate SOAS functions were selected in the following areas: Situation Assessment (SA), Tactics Planner (TP), and Command Interface (CI). The selection of functions has been an iterative process, depending on the following factors: the ability to develop programming techniques to provide the function, the likelihood of a suitable method of communicating the function to attack center personnel, the availability in the technology base of similar functionality, and the ability to provide appropriate scenario situations to exercise the functions. As the functions have been attempted, and in many cases tested, modifications in the functions' selection and allocation have occurred.

For example the Phase I program initially included the definition and design of a Ship's Operations (SO) subsystem. This subsystem was intended to assist the Commanding Officer (CO) by monitoring, alerting, and advising him about the schedule for routine ship functions and operations, changes in conditions affecting current operations, and the status of on-board systems and equipment. In order to streamline the scope of LASC's SOAS program, a decision was made not to implement SO as a primary subsystem. However, the architecture developed during the Phase I effort will allow SO functions, as well as other desired enhancements, to be incorporated easily over the life of the system.

Subsystem Specification - Beginning with the functional areas defined already in the Pilot's Associate Program, the allocation of functions resulted in the selection of separate subsystems with similar names.

Technology Assessment - Throughout the program, continual assessment of past and on-going research in the areas of planning, assessment, man-machine interface, and real-time execution of knowledge-based systems has been conducted.

Prototyping and Incremental Integration - Beginning with the program kickoff, limited functionality software was delivered periodically for demonstration in Lockheed's Artificial Intelligence Development Laboratory (AIDL). This prototyping and incremental integration served several purposes:

- It validated the subsystem interaction specification.
- It provided for early evaluation of system performance.
- It tested the completeness and consistency of system design.
- It provided a measure of the maturity of subsystem designs.
- It exposed specific simulation and test requirements.
- It exercised and validated the integration methodology.
- It provided early insight into system response-time problems.

Software Integration Management - In order to maintain a consistent design specification while, on the one hand, system requirements were evolving and, on the other, prototype software was being built and delivered, several tools were developed to represent and maintain the system and subsystem specifications. These include the following:

- A SOAS-to-SOAS-to-Simulation Interface Control Document (ICD)
- A System Plan-Goal Dictionary

These specifications were maintained and published by Lockheed and have evolved greatly since the beginning of the program.

Demonstrations and Evaluations - As the SOAS system and simulation environment matured, it was possible to demonstrate and evaluate specific functions at both the subsystem level and the SOAS system level. As subsystems were delivered, scenario segments constructed to exercise each required function, and much dedicated integration time was spent evaluating the evolving system.

1.4 TEAM MEMBERS AND RESPONSIBILITIES

Lockheed selected a team of four subcontractors to assist in the development of the Submarine Operational Automation System. Team member selection was based on capability in each of the essential areas, including AI tools and techniques, domain knowledge, and software development expertise, as well as the willingness to share technology among other team members. Each team member played a unique role, as shown in Figure 1-1. Rather than locating all software development tasks at one site, all knowledge engineering at another, and all domain expertise at another, the Lockheed approach has been to give the responsibility for leadership of entire subsystems to individual team members, with other team members supporting as necessary. As a result, a great diversity of tools and techniques has been explored, and a great deal of expertise has been transferred.

	Subsystem			
Subcontractor	**SA**	**TP**	**CI**	**SM**
Lockheed Aeronautical Systems Company	L,S,T	D		L,S,D,T
Search Technology,Inc.	D	D	L,S,D,T	
ISX Corp.		L,S,T		
Presearch, Inc.	S,D,T	D	T,D	D
Kapos Associates	D	D	D	D

Key:		
L	-	Lead (System Requirements, Top-Level Design)
S	-	Software Design and Development
D	-	Domain Expertise
T	-	Technology Support

Figure 1-1. Team Member Responsibilities

1.5 KEY ACCOMPLISHMENTS

The Lockheed Submarine Operational Automation System Phase I effort has been successful, mixing higher-risk theoretical research with down-to-earth software development, integration, and testing. From the Pilot's Associate program and the SOAS Phase I effort have come a mature system framework, a large quantity of working software, a proven method for expanding system capability, and a clear path to develop real-time performance. Highlights of the Phase I results are summarized below.

1. A **system design and development methodology** that includes configuration-controlled data flow descriptions, a subsystem interface control document, response time analyses, test procedures, a central executive subsystem with scripted versions of all other subsystems, and a system-level knowledge structure, called the Plan-Goal Dictionary, through which all planning is coordinated.

2. A **proven prototyping** methodology and successful prototype integration exercises.

3. An **environment for evolution of working prototypes**, which allows incremental integration of the SOAS system, experimentation with multiple methods within each subsystem, flexible simulation and test environment with displays, and a network of machines for developing and testing different methodologies.

4. A **documentation strategy** which emphasizes frequent, informal reporting, consistent with the continual experimentation and system evolution.

5. The **sharing of technology** through contacts with other related programs, including DARPA Technology Base efforts such as ABE Real-Time, Reasoning with Uncertainty Module and the Pilot's Associate Program.

1.6 SOAS PHASE I DEMONSTRATIONS

At the conclusion of Phase I, the LASC team conducted demonstrations which incorporated the specific features identified as goals for the SOAS program and, additionally, showed the promise of attaining real- time performance. Two types of demonstration were successfully performed: subsystem demonstrations which featured component technologies and a display demonstration which presented LASC's concepts for man-machine displays and interfaces.

1.6.1 Subsystem Demonstrations

These subsystem demonstrations covered the range of functionality needed to enhance the tactical and operational performance of advanced submarines.

- **The Tactics Planner subsystem** was demonstrated in a stand-alone mode on a Symbolics machine and used data files which emulated the input data it would receive from other components of SOAS. The TP demonstration illustrated the meaningful decision support which can be provided to the Commanding Officer in tactical situations. The subsystem showed the ability to reason in the tactical domain with uncertain and incomplete information, to generate feasible tactical options, and to communicate the results to the Commander in a timely manner. The TP subsystem was implemented using Kadet, a state-of- the-art skeletal planning tool which models operational situations and provides the TP with a conflict-resolution capability.

- **The Command Interface subsystem** was also demonstrated in a stand-alone mode in a manner identical to the Tactics Planner. The CI subsystem successfully performed its primary function of interface management, which involves decoding inputs and encoding outputs, selecting information to be displayed, and configuring controls to meet the Commanding Officer's current and short-term plans. Included were features to accept and reject SOAS-generated plans. In addition, CI's intent inference function was demonstrated. This latter function, which supports interface management, processes submarine state information in order to infer what the CO is trying to accomplish. The results of the intent inference function are vital to the Command Interface subsystem in integrating the CO's plans and goals with those proposed by SOAS.

- **Situation Assessment and System Manager** were integrated with the simulation environment to show the effectiveness and feasibility of these two primary subsystems. Situation Assessment monitored the external environment for the occurrence of contacts. SA evaluated the capabilities, intent, anticipated responses, and lethality of contacts; provided track data about them; computed the closest point of approach; and predicted engagement outcomes. The System Manager, on the other hand, successfully controlled all interactions, maintained the system database, and coordinated activities between the simulation and the SA subsystem.

- **The display demonstration** was conducted in the theatre adjacent to LASC's Artificial Intelligence Development Laboratory. It featured the System Manager subsystem integrated with the simulation environment and a Display Generator. SM received contact reports from the simulation, processed the data, and sent format commands to the Display Generator, which produced displays. Types of displays which were shown included periscope depth operations, contact classification clues, weapon effectiveness, search evaluation, measures of effectiveness for engagements, and general ownship status.

The following eight figures contain sample Symbolics screens and workstation displays as well as the physical layout of LASC's laboratory and other features of the Phase I demonstrations.

Figure 1-2, Situation Assessment screen - The SA-simulated fusion module has helped classify a track (No. M002) and SA has actually calculated the predicted counterdetection envelope in the center display. Displayed on the bottom left is the pertinent data from the SA Track database. One external event of interest is CPA, which is being monitored by SA. At the bottom middle of the screen is a message indicating that there is a danger of collision.

Figure 1-3, Command Interface screen - Shown is a portion of the SOAS Plan-Goal Graph related to Periscope Depth Operations. On the middle right side of the screen are graphics which would be displayed in an advanced submarine to assist the CO in P/D Ops.

Figure 1-4, Tactics Planner Subsystem - On the monitor is an example of the SOAS Plan and Goal Graph which is being used by the developer to insert knowledge into the Route Planning and Navigation module. On the right is a diagram of the TP architecture for future phases of SOAS.

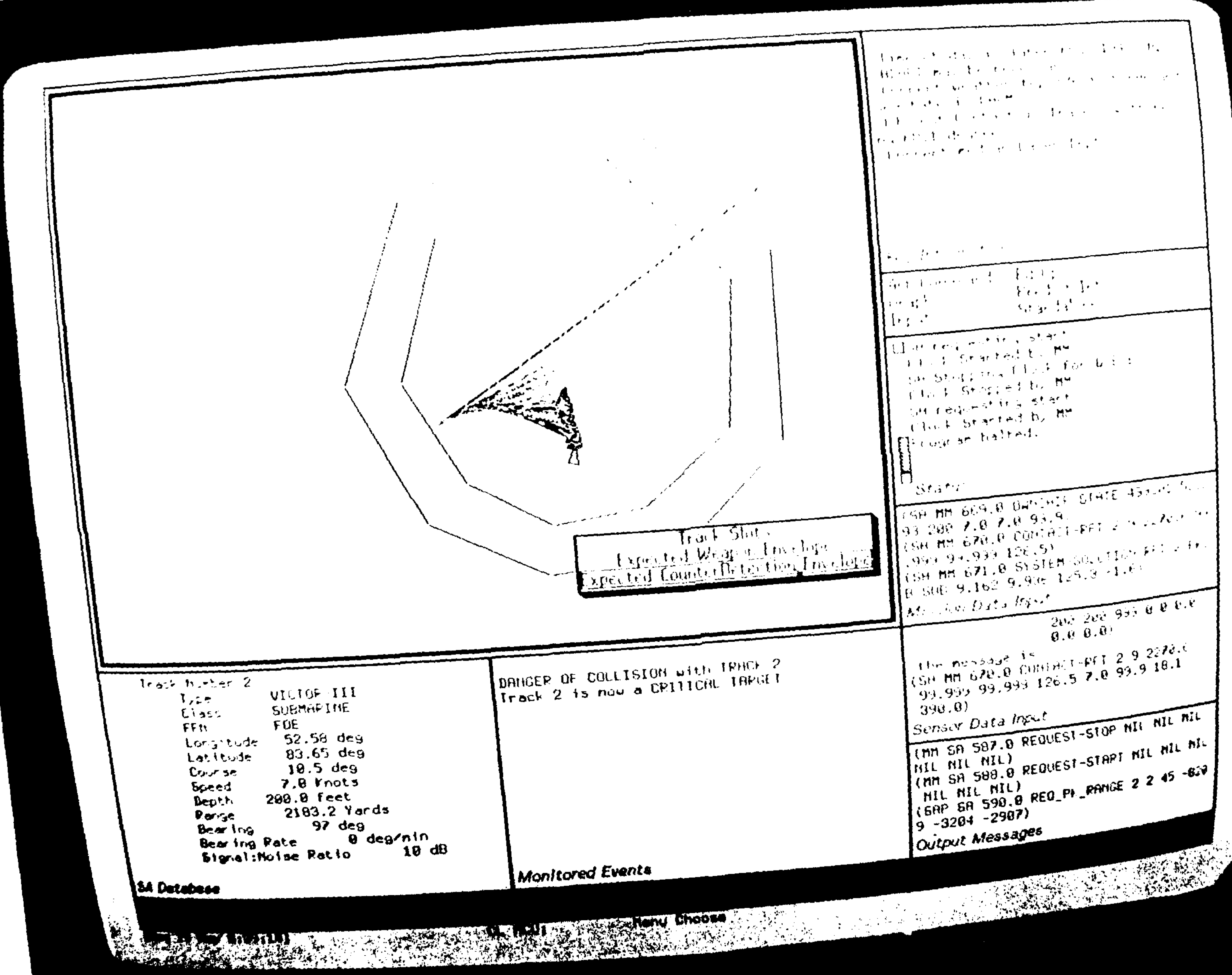
Figure 1-5, SOAS Theater, AI Development Laboratory - Shown on the left are the two monitors and a touch panel used to present SOAS displays for a generic command-level officer. On the right side is a large "futuristic" display which would be used to guide the attack party. The display can show tactical situations as well as Command intent.

Figure 1-6, Search Evaluation Display - Shown on the display are the horizontal and vertical probability of detection zones for a search evaluation. These zones are significant when trying to evaluate the success of a search.

Figure 1-7, Generic Ownship Display - In the upper right corner is a prototype of a graph for speed-vs-depth for a cavitation curve. The bottom display shows the difference in the ship's trim since the last P/D operation. The right side contains a prototype display of ownship information.

Figure 1-8, Predictive Engagement Displays - The right display presents the probability of hit for an unalerted target with the expected, worst case, and best case outcomes. The displays on the right illustrate the probability of detection for a vertical salvo and a horizontal salvo against a target.

Figure 1-9, Likelihood Ratio Tracker (LRT) Display - Using the LRT, a display can be generated which represents the relative probabilities of each possible location and course interval for a track. Relative color weighting makes this an important aid in making planning decisions.

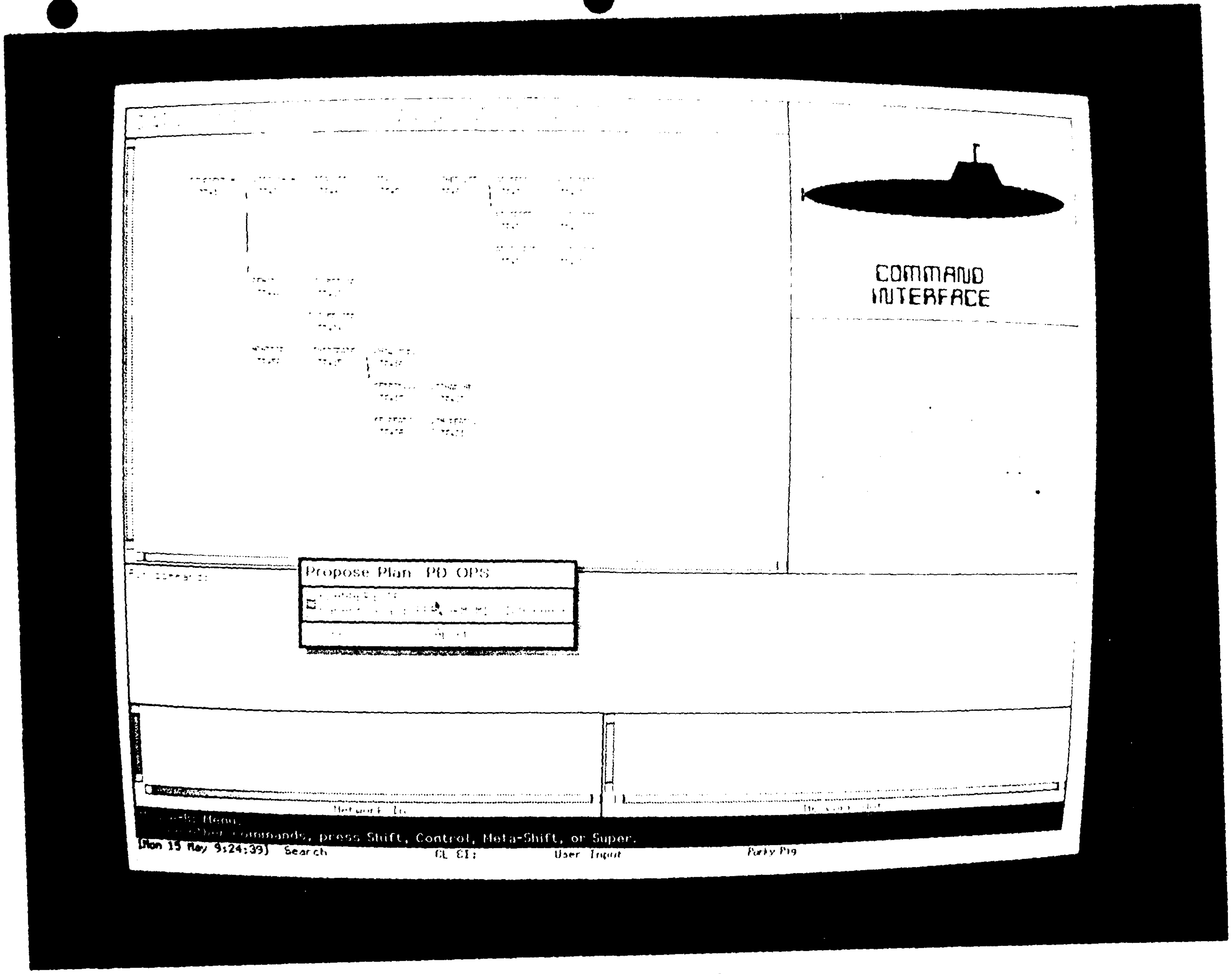


```
Track Number 2
    Type          VICTOR III
    Class         SUBMARINE
    FFN           FOE
    Longitude      52.58 deg
    Latitude       83.65 deg
    Course         10.5 deg
    Speed          7.0 Knots
    Depth        200.0 Feet
    Range          2103.2 Yards
    Bearing           97 deg
    Bearing Rate        0 deg/min
    Signal:Noise Ratio       10 dB
SA Database
```

```
DANGER OF COLLISION with TRACK 2
Track 2 is now a CRITICAL TARGET

Monitored Events
```

```
[illegible]

[illegible]
Clock Started by MM
[illegible] Clock for [illegible]
Clock Stopped by MM
SA requesting start
Clock Started by MM
Program halted.
Status

(SA MM 669.0 OWNSHIP STATE [illegible]
93 200 7.0 7.0 93.9
(SA MM 670.0 CONTACT-RPT 2 [illegible]
(SA MM 670.0 CONTACT-RPT 2 [illegible]
.999 99.999 126.5)
(SA MM 671.0 SYSTEM SOLUTION RPT 2 [illegible]
0 SUB 9.162 9.906 125.3 -1.6)
[illegible] Data Input
                 200 200 999 0 0 0.0
                 0.0 0.0)
 the message is
(SA MM 670.0 CONTACT-RPT 2 9 2270.0
 99.999 99.999 126.5 7.0 93.9 10.1
 390.0)
Sensor Data Input
(MM SA 587.0 REQUEST-STOP NIL NIL NIL
NIL NIL NIL)
(MM SA 588.0 REQUEST-START NIL NIL NIL
 NIL NIL NIL)
(GAP SA 590.0 REQ_PT_RANGE 2 2 45 -820
9 -3204 -2907)
Output Messages
```

Figure 1.2



[illegible] Menu.
[illegible] commands, press Shift, Control, Meta-Shift, or Super.
[Mon 15 May 9:24:39] Search CL CI: User Input Porky Pig

Figure 1-3

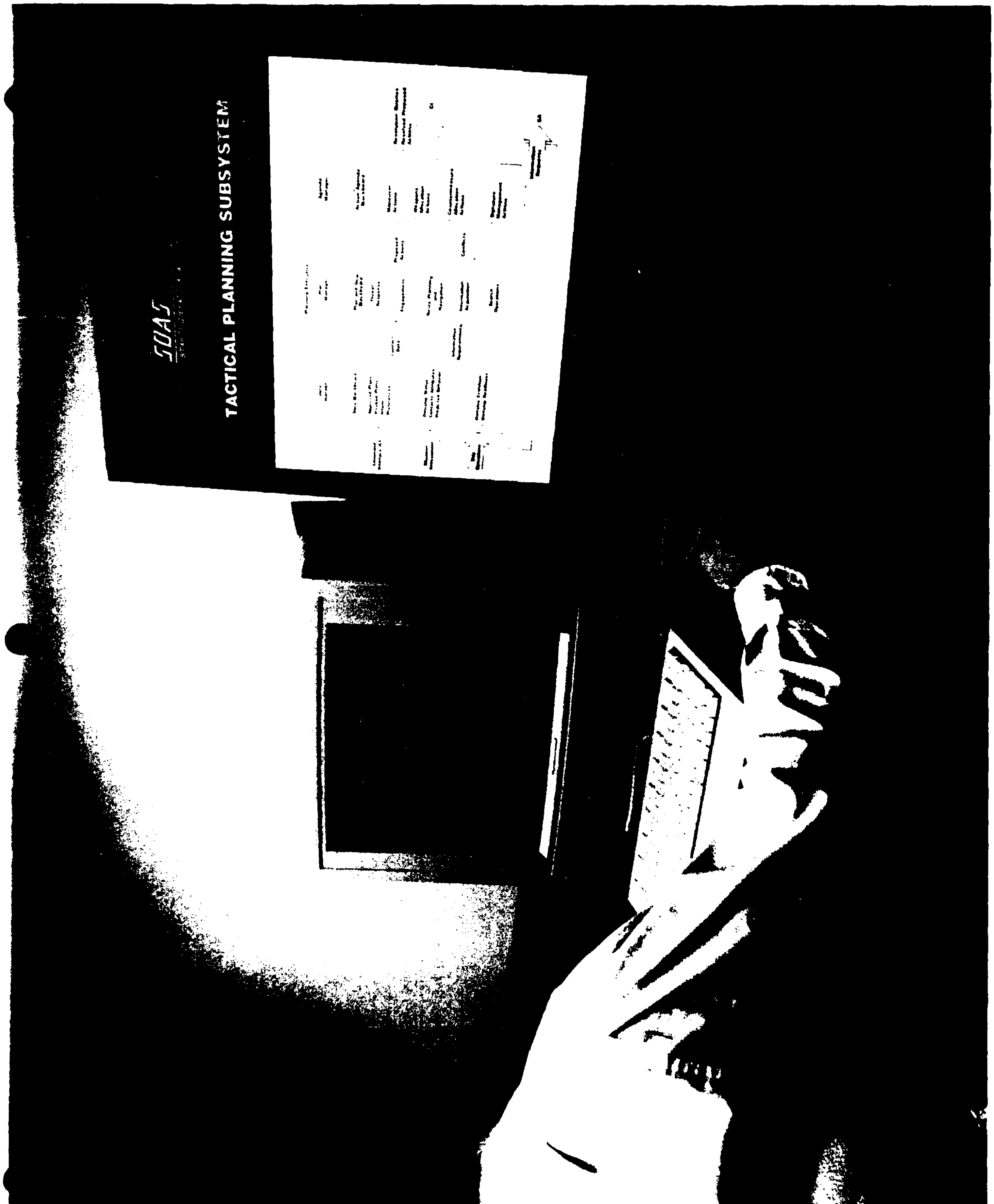
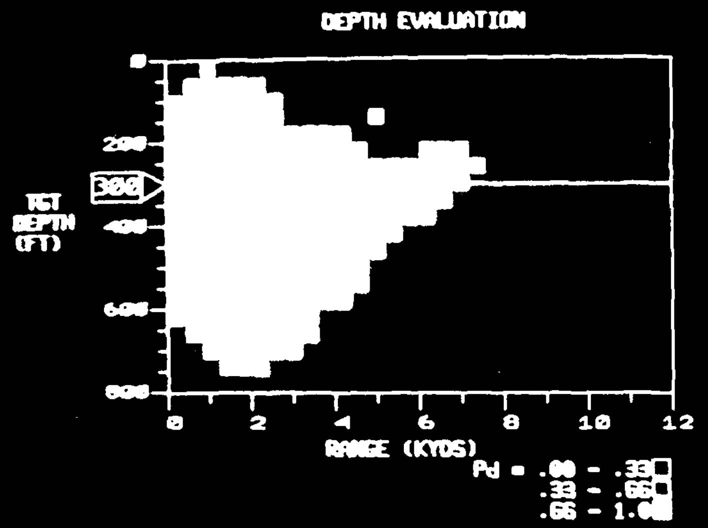
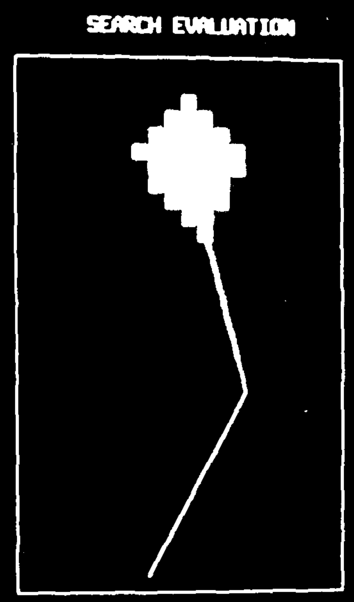
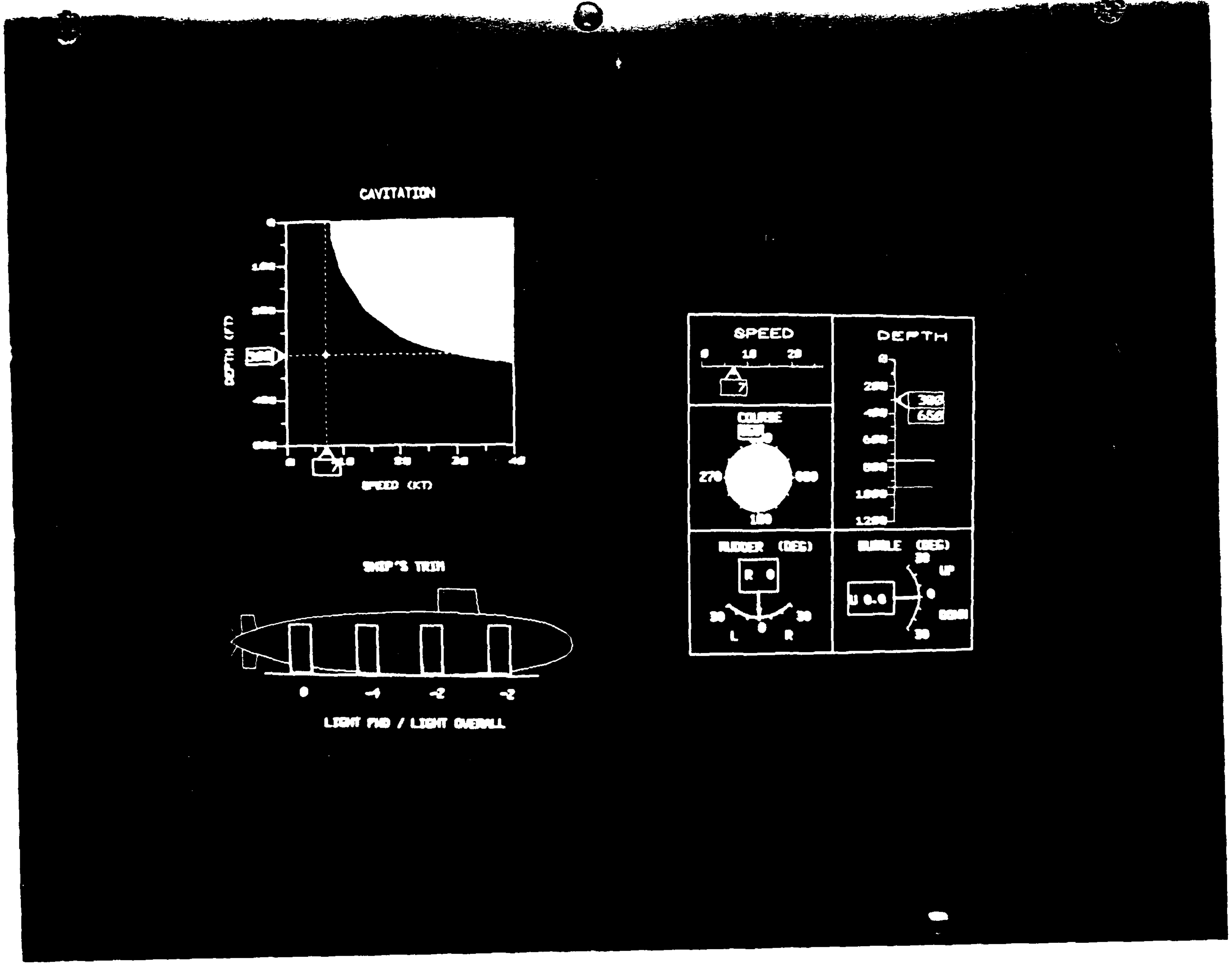


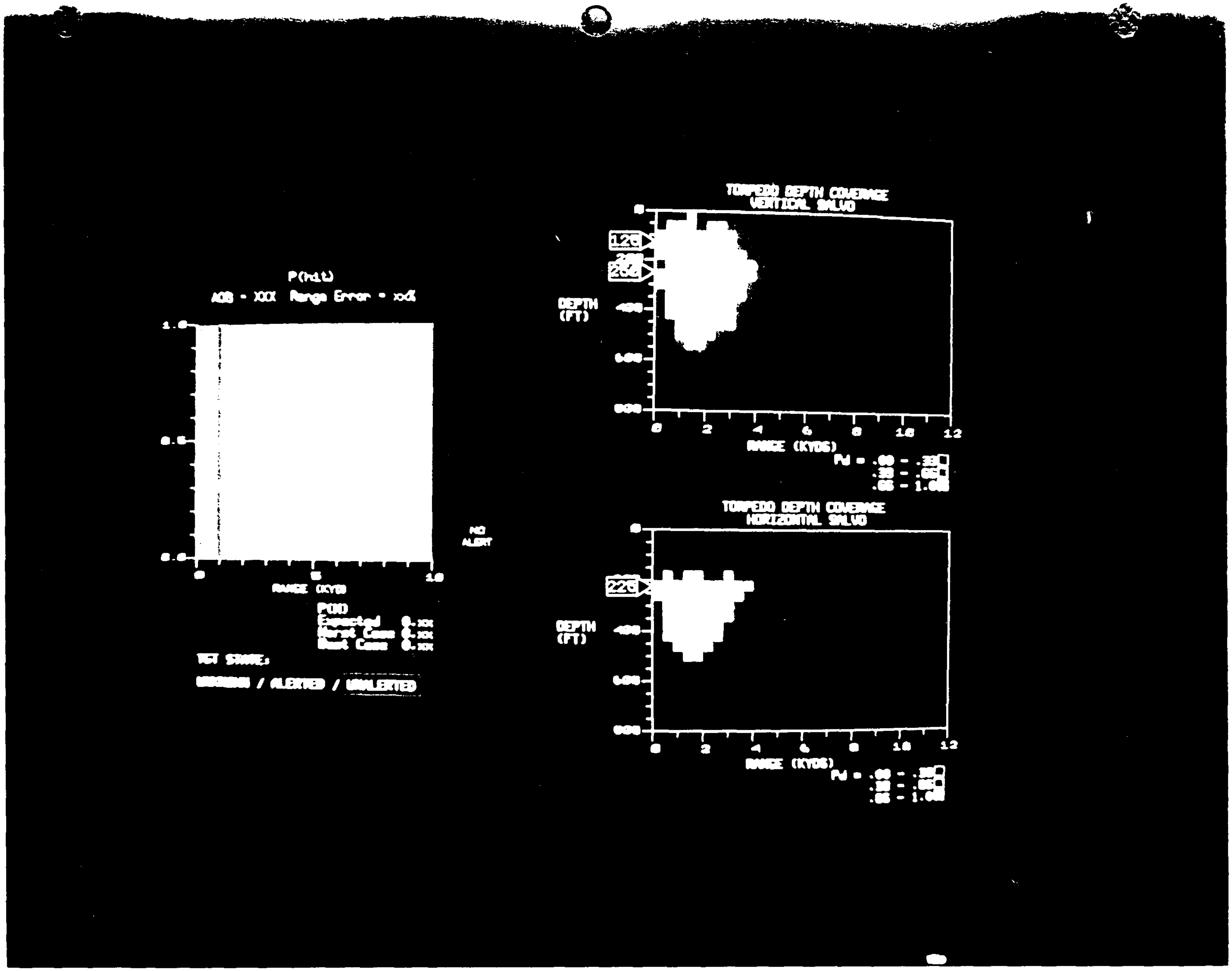
Figure 1-4

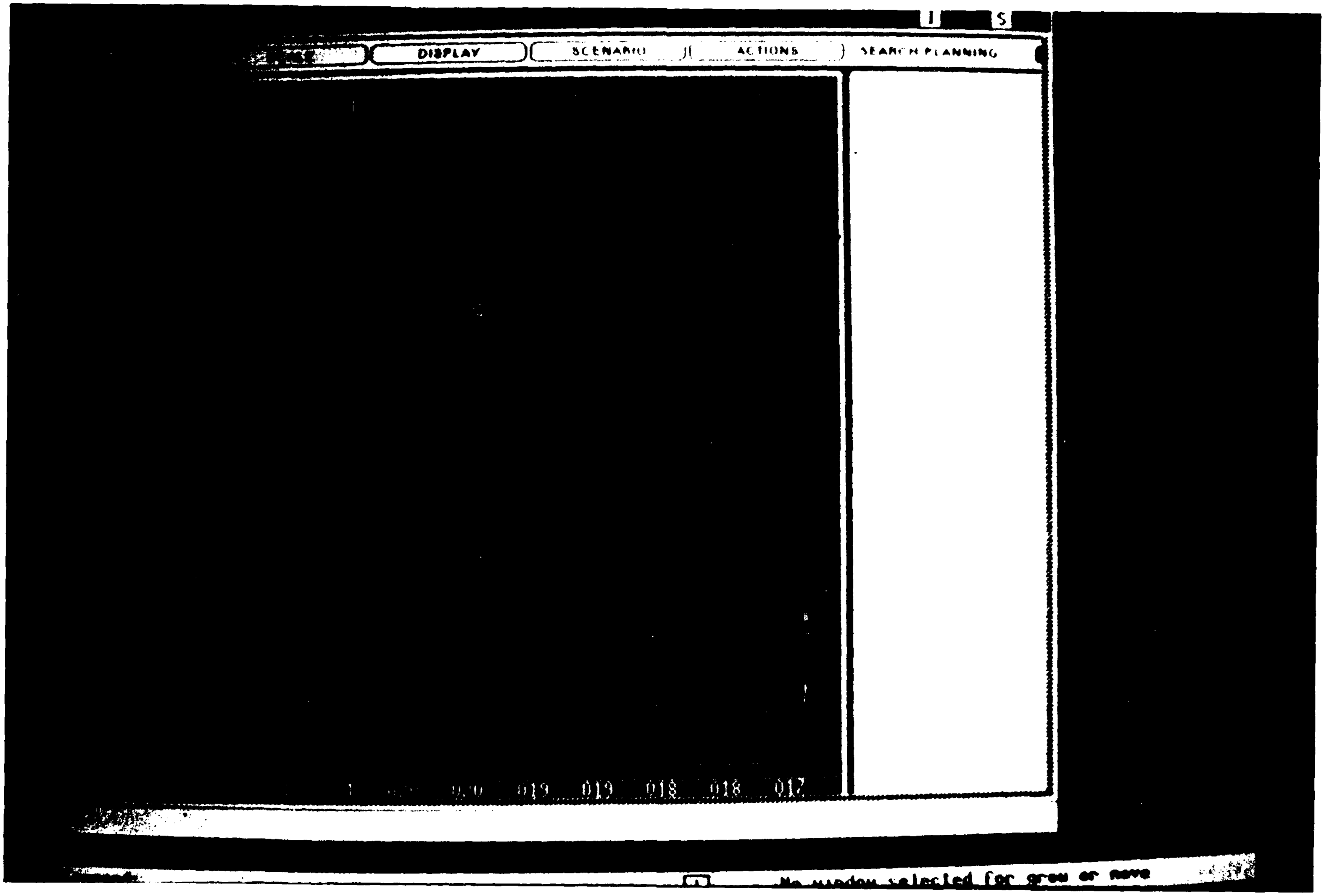
Figure 1-5


DEPTH EVALUATION
TGT DEPTH (FT)
0
200
300
400
600
800
0
2
4
6
8
10
12
RANGE (KYDS)
Pd = .00 - .33
.33 - .66
.66 - 1.00


SEARCH EVALUATION


CAVITATION
DEPTH (FT)
SPEED (KT)
SHIP'S TRIM
LIGHT FWD / LIGHT OVERALL
SPEED
DEPTH
COURSE
RUDDER (DEG)
BUBBLE (DEG)
UP
DOWN
L
R
270
180


P(hit)
AOB = XXX Range Error = xx%
1.0
0.5
0.0
0
5
10
RANGE (KYD)
NO ALERT
POD
Expected 0.xx
Worst Case 0.xx
Best Case 0.xx
TGT STATE:
UNKNOWN / ALERTED / UNALERTED
TORPEDO DEPTH COVERAGE
VERTICAL SALVO
DEPTH (FT)
120
200
250
400
600
800
0 2 4 6 8 10 12
RANGE (KYDS)
Pd = .00 - .30
.30 - .65
.65 - 1.00
TORPEDO DEPTH COVERAGE
HORIZONTAL SALVO
220



No window selected for grow or move

Figure 1-9

2.0 SUBMARINE OPERATIONAL AUTOMATION SYSTEM

2.1 CONCEPT OF OPERATION

Since the advent of machines, the role of the human has become increasingly to manage the machines and monitor their performance. In the combat environment, however, the human must not be required to manage and monitor all submarine systems. Rather, his intent and actions should be monitored and understood by the machine so that he can be truly supported in assessment, planning, and plan execution during combat, tracking, and normal underway operations.

The command centers of future submarines must face an increasingly technically sophisticated enemy, in superior numbers. While advances in submarines, sensors, weapons, and other systems address the problem, they also increase the amount of data and information which the control room/attack center personnel must assess. In addition, the management of these systems creates an unacceptable workload at critical mission times. The results are, potentially, more submarine losses or mission aborts through loss of situational awareness, and fewer enemy destroyed because the submarine and its capabilities are not used to their maximum capability and thus offensive opportunities are lost.

2.1.1 Operational Considerations of Full-Scale System

The SOAS challenge is to assist the submarine commanding officer in the tactical execution of his mission. This can be accomplished with a combination of decision aiding, data integration, information display, and problem focus. The submarine decision maker is typically faced with an excess of data, much of which is ambiguous, from which he must focus on the critical issues and make tactical decisions. SOAS will integrate and structure the data to provide information, not data, to the decision maker; it will help the decision maker focus on the critical issues through its unique concept of "dominance"; it will alert the decision maker if routine conditions or events become constraints; it will provide tactical planning and answer "what if" questions, and it will anticipate future requirements by drawing inferences from ownship and enemy options.

Routine Monitoring and Alertment

A submarine operates in an inherently hostile environment where water pressure (depth), buoyancy (control of water inside the pressure hull) and the ship's atmosphere (life support) must be routinely monitored and controlled. Although potentially dangerous, experience has made control of this environment routine. The submarine decision maker must know his environmental situation and be prepared to react instantly to problems, but he cannot afford to devote any time to these routines. This dichotomy begs for a SOAS system that will monitor routine functions, alert the decision maker to problems, and recommend action thereby "allowing" the decision maker to concentrate on the more important problems with confidence that he will not jeopardize ship safety or mission success by "neglecting" routine functions.

The Need For Information Not Data

A major characteristic of submarine combat system management is data excess. The submarine routinely operates with multiple contacts and may be required to simultaneously engage several targets while conducting operations in multiple warfare areas. For example, a submarine may find it necessary or desirable to launch torpedoes while conducting a Tomahawk strike. The worsening problem of data excess must be managed by converting data into concise, comprehensive information because the data excess problem is endemic to the submarine's situation. Required to rely solely on passive acoustic data for most of its information, submarine combat systems have developed several methods of manipulating the acoustic data to provide a maximum amount of information. This approach is appropriate and necessary but does result in inconsistent estimates and tends toward data excess.

Data Integration

Submarines are making increasing use of off-board data. The risk involved with acquiring such data demands its integration with other data for effective use. The alternative is data pollution resulting in multiple target tracks from single contacts, unsuccessful attempts at vectored intercepts, and an inability for the submarine to participate in combined arms operations. Whether the submarine is trying to use remote, space sensor data and information from an on-board acoustic array to perform an independent mission or is trying to locate and provide targeting information to a mined party launch platform, data integration will be a primary requisite for success. This is another problem that will increase in complexity in the near future.

Adaptation of Machine Into a Team

Submarine operational automation is a more demanding challenge than traditional man-machine interface problems because the submarine system must integrate into an efficient team of personnel. Experience has taught that attempts at one-for-one replacement of an attack center individual by a machine is inefficient and ultimately disruptive to the team. The adaptation of operational automation must be accomplished because machines simply do some things better than individuals and because future submarine combat systems may be so large or distributed that the decision maker may be unable to visually observe the workings of his team. Operational automation is a concept whose time has arrived for submarines but whose implementation must be carefully orchestrated to insure improved effectiveness and continuation of the important submarine team concept.

Extendable Methodology

The roles, missions, and tactics of submarines could change dramatically over the next 20 years. Submarines will very likely control autonomous vehicles, but may also be required to operate in the undersea terrain or assume the task of destroying ballistic missiles and satellites. It is, therefore, of utmost importance that submarine operational automation systems be extendable to those new roles, missions, and tactics. Changes should be anticipated and new systems cannot be allowed to encourage stagnation because of their rigidity.

Resolution of Ambiguity

There are two distinct types of ambiguity that permeate the submarine combat control problem. One involves the assembly of partial information into a complete picture (has own ship been counter-detected) and the second resolution of conflicting data (determine target range from a series of inconsistent range estimates). Both are important and must be resolved.

Information Display

Information display, not only how it is displayed but what is displayed, will have a significant impact on future submarine capability. As the likelihood increases that a submarine will simultaneously conduct several tactical operations (e.g., control a mine field penetration and Tomahawk strike simultaneously, or control a scout MMV while operating under the Arctic ice and attacking a Soviet SSBN), the very substance of the information presented to the decision maker, as well as it formatting for rapid assimilation, will become increasingly important. It is also likely that the decision maker's movement may take him outside of the visual range of a single display because of the size and arrangement of future combat centers/control rooms.

2.1.2 Solution Statement

It has become evident that the techniques of expert systems offer promise in several areas of information management and decision aiding. However, most of the successful expert systems that have been developed are intended for use in a static, single workstation environment. Only recently have systems begun to appear that have been interfaced with other computing systems, and perhaps even other AI systems, and have been designed to respond to rapidly changing problem domains.

The Submarine Operational Automation System program is aimed at demonstrating the feasibility of applying the techniques of AI, and in particular knowledge-based systems, to the problems faced by control room/attack center personnel. To do this it is necessary to accomplish the following:

1. Demonstrate that individual systems can be developed whose functions support the control room/attack center personnel in critical tasks. This includes demonstrating that the relevant knowledge can

be collected and represented in an accurate way, and that it can be used reliably to reason about the incoming data available onboard the submarine.

2. Demonstrate that the systems can work in conjunction with other submarine systems, such as sonar and fire control systems, in the dynamic underwater environment.

3. Demonstrate that the information produced by knowledge-based systems can be communicated to the key control room/attack center personnel in a timely manner, and verify that the functions performed by the system can be done in such a way that they are coordinated with all crew actions and intent and so that no command authority was lost.

4. Demonstrate that the response-time of the system in each functional area is adequate for use under the harshest conditions.

In order to demonstrate the feasibility of a command-aiding system such as SOAS, the system first had to be defined. The program outlined by DARPA emphasized thorough system analysis and systems engineering, but also required the demonstration of feasibility by means of early, working system demonstrations as primary program goals. In connection with these requirements, the goals of the Submarine Operational Automation System program involve the creation of prototype systems with increasing capability. The system performance goals for the Phase I system demonstration are shown in Figure 2-1.

2.1.3 <u>Subproblems</u>

In decomposing the top-level problem statement of Section 2.1 into the solution structure described in Section 2.2, the subproblems described in the following paragraphs were identified. The material in Sections 3.0 through 5.0 describe the solutions to these subproblems as they were evolved and implemented during Phase I prototype development. It should also be noted that the changing system design during subsequent phases of the SOAS contract may radically alter the character both of these subproblems and of the solutions provided.

<u>Mission and Submarine Definition</u>

A scenario was specified as the basic framework in which to develop the SOAS mission requirements. Considerable care was required in defining mission events, submarine performance, weapons and sensors, and display requirements in order to provide a scenario with sufficient detail to be a useful reference for system functionality development. Adaptation was also required to generate situations which stressed the relationships among SOAS subsystems while retaining operational realism. Section 5.1 summarizes the basic scenario used for system test and the Phase I demonstration.

The definition of an out-year submarine was a problem from the following points of view:

SITUATION ASSESSMENT

Knowledge/ Functions	Assess lethality of a subsurface contact Assess intent of a subsurface contact Assess target value for all contacts Assess probability of counterdetection, kill and counter-kill
Interactions	Establish dynamic monitoring criteria for plan generation and monitoring

TACTICS PLANNER

Knowledge/ Functions	Plan and monitor an engagement of a subsurface contact Plan and monitor a response to a manuvering contact
Interactions	Set constraints for route planning based on tactical engagement requirements

COMMAND INTERFACE

Knowledge/ Functions	Determine and track command intent during an engagement mission, including implicit plan approval Dynamically manage display configuration
Interactions	Detect/transmit plan approval of tactical and route plans

Figure 2-1. Demonstration 1 System Performance Goals

- existence and availability of information
- security level of the information
- quantity of data required

The first approach was to develop a "generic submarine" based on unclassified estimates of the capabilities of submarines in the late 1990's. However, the level of detail necessary to support the needs of the Situation Assessment and Command Inteface subsystems made this very difficult to accomplish within the allocated resources. In many cases, the data needs of SOAS have exceeded the ability of the program to provide information.

Multiple (Dissimilar) Subsystems

This subproblem is a natural consequence of the freedom given to subsystem developers to choose tools and environments best suited to their particular problem domains. Section 3.0 discusses the integration strategies established to ensure that the dissimilar subsystems, while optimized for solving their own particular problems, were communicating correctly with appropriate information exchange paths.

Evaluation of System and Subsystem Functions

Despite diligent searches of conferences, papers, and academic institutions, there has been little comprehensive work uncovered which addresses the global issues related to knowledge-based system evaluation and validation. Clearly, this must be addressed in the phases of the SOAS program which precede Full- Scale Development. The Lockheed team is taking aggressive steps on related programs to develop methodologies suitable for knowledge base validation.

Some Lockeed Independent Research and Development work in measures of effectiveness has been initiated with the objective of defining methods by which the value of systems such as SOAS can be quantified.

Technology Maturity

Each of the various Lockheed divisions and subcontractors involved in SOAS came to the program with experience, tools, and working application software ready to apply to the problem. The initial product was, therefore, remarkably mature. The success of the Phase I Demonstration has now brought increasing pressure to enhance the maturity of the software tools, the design and development methods, the application code, and the hardware on which it runs. The future maturation paths will be described in the Lockheed System Development Plan for the remaining phases of the project.

2.2 SYSTEM FUNCTIONS AND COMPONENTS

The functional architecture developed for the Phase I Submarine Operational Automation system is shown in Figure 2-2.

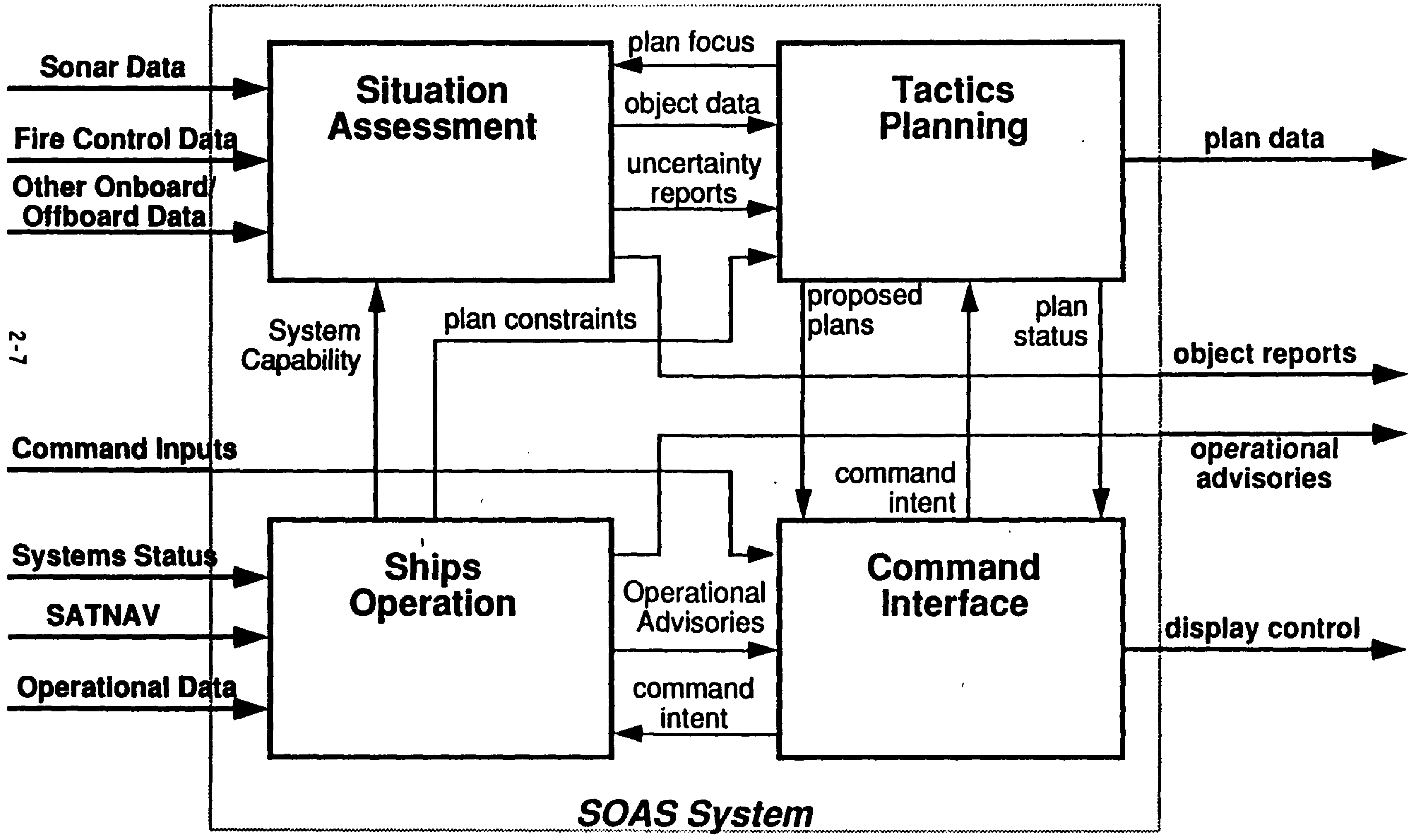


Figure 2-2. System Functional Flow

In order to provide assistance while avoiding display complexity, SOAS has a highly intelligent interface between the console displays and all the key supporting members of the control room. In Phase I this interface was developed for use by the Commanding Officer (CO) or the approach officer. It is capable of estimating the command's needs and priorities and managing the presentation of information to them. It can be personalized to the CO's preferences for default actions and decisions, the level of authority assigned, and the levels of information display preferred. Within this context, SOAS can provide unprecedented levels of automated decision support to command-level personnel. Figure 2-3 shows the Command Interface (CI) displays that the full-scale development program will address. The CI functionality and processing methods implemented in Phase I are described in detail in Section 4.1 of this report.

The Phase I SOAS provided assistance to command-level personnel in assessing the external situation. It used the output from simulated acoustic sensors and the fire control solution to determine intent, lethality, and vulnerability of each of the enemy units. The Situation Assessment subsystem of SOAS also assessed the expected engagement outcomes for the given scenario on request from the CO and the Tactics Planner subsystem (TP). This assessment included Measures of Effectiveness for engagement planning, weapon modeling, and environment modeling. The SA functional flow is show in Figure 2-4 and further definition of the functionality and processing is defined in Section 4.3.

The Phase I SOAS Tactics Planning subsystem provided suggestions for dealing with current tactical events. Included were recommendations for TMA, attack plans and weapons commitments. The TP subsystem demonstrated the ability to defer the commitment to plan based upon the lack of information available. The planner featured the ability to show "what-if" reasoning by showing how perceived values could be changed and planned about. The functional flow of TP is shown in Figure 2-5 and further definition of the functionality and processing is defined in Section 4.2.

The Ship's Operations (SO) subsystem was explored but not demonstrated in Phase I. The goal of SO was to help the crew with normal operations so that they could concentrate on higher-level decision making. Its functions included monitoring and assisting in navigation and maintentance of track/PIM, monitoring ship's signature, having current status available, maintaining ultra-quiet conditions of equipment, providing for alertment of failures, estimating impact of operational capability and recommending work-around alternatives, alerting of events and changes in the local environment, and monitoring routine functions/operations to reduce the data overloads.

One of the primary functions of the System Manager (SM) subsystem was to manage the interface among the external programs and SOAS. In Phase I, the external program was the simulation environment. Another major function, the strategy for integrating and controlling the various functions of SOAS, was based on a blackboard architecture, which has been used in many AI-based systems. The System Manager blackboard not only allows all subsys-

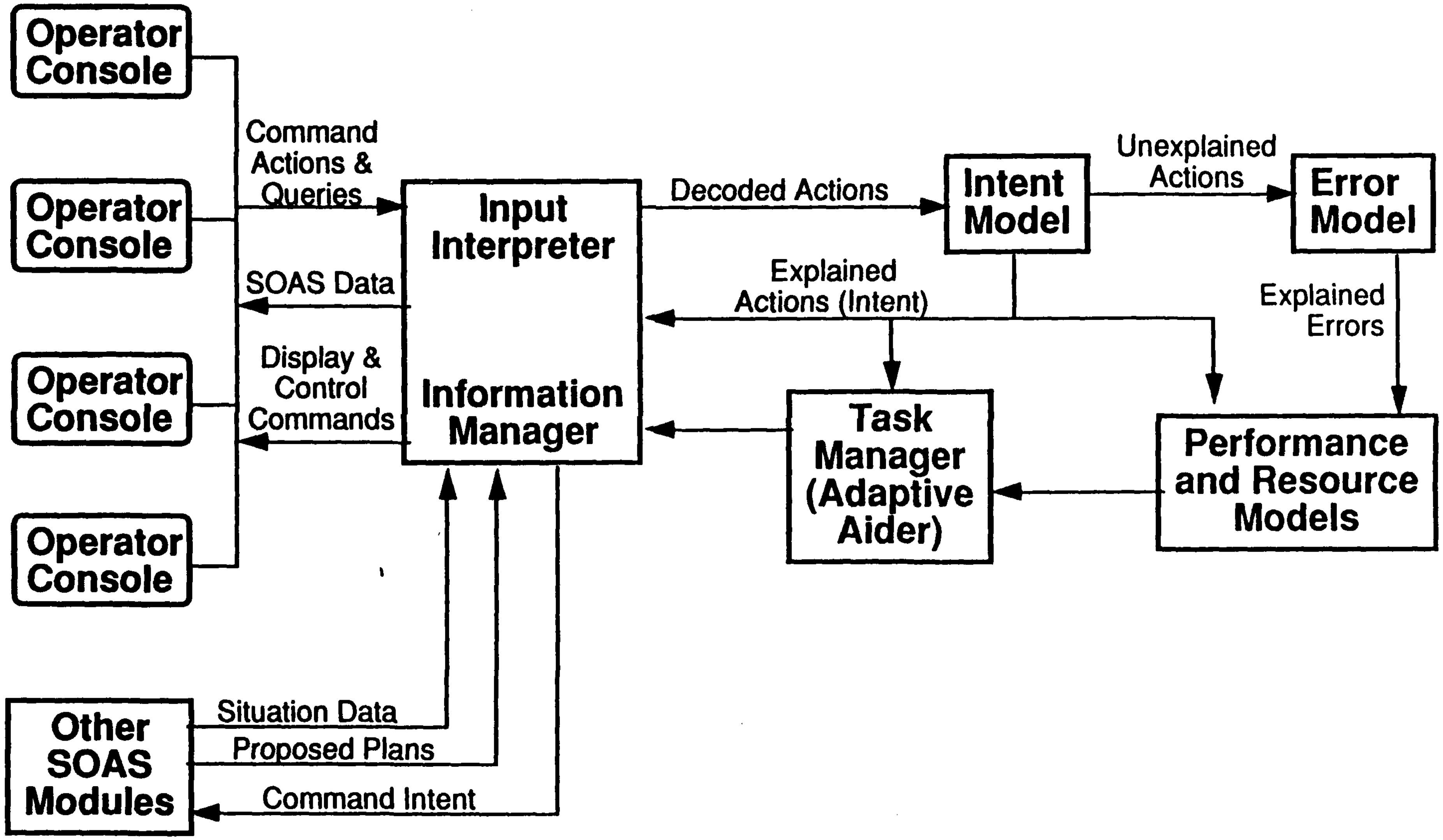


Figure 2-3. Command Interface Functional Flow

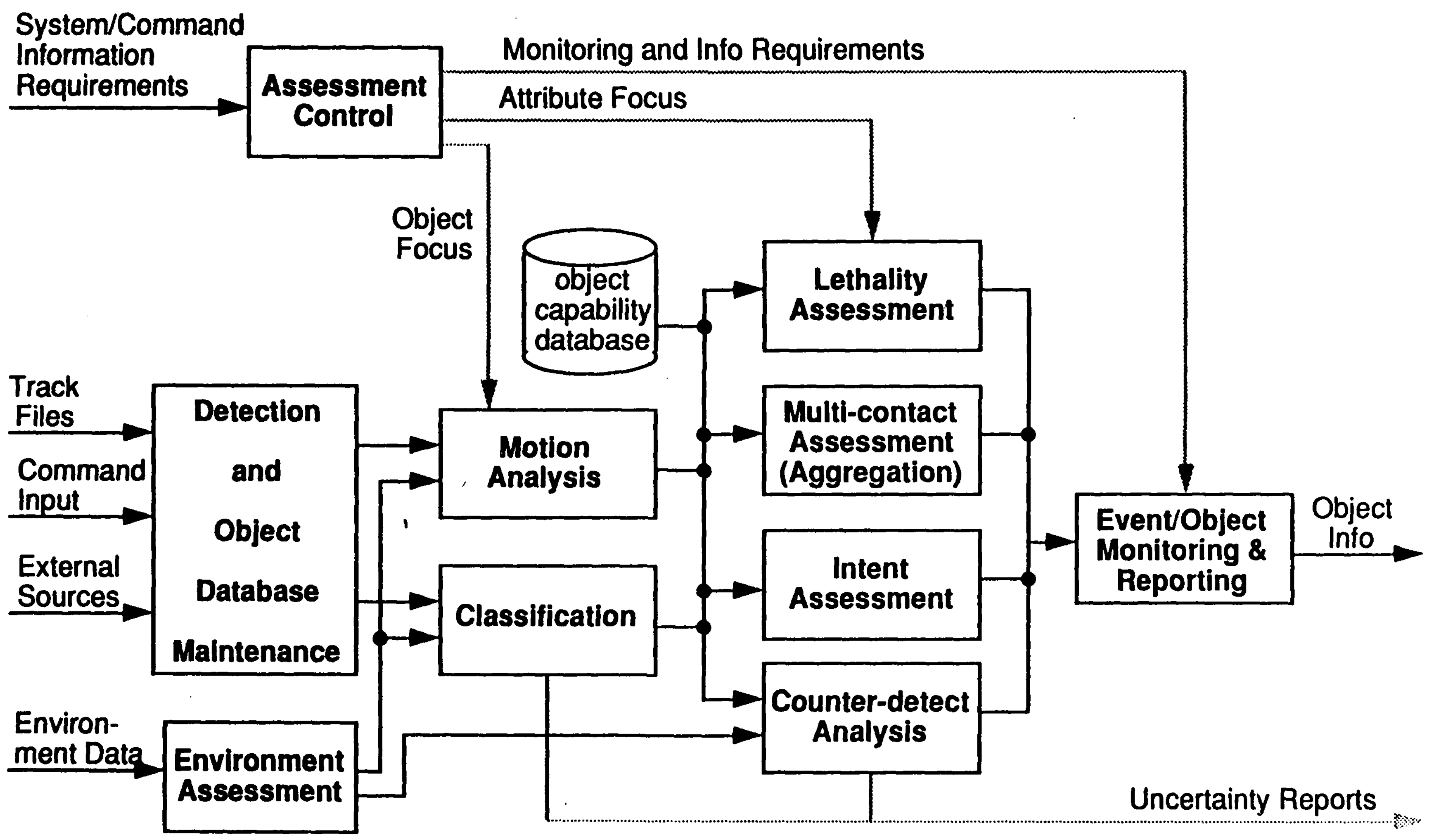


Figure 2-4. Situation Assessment Functional Flow

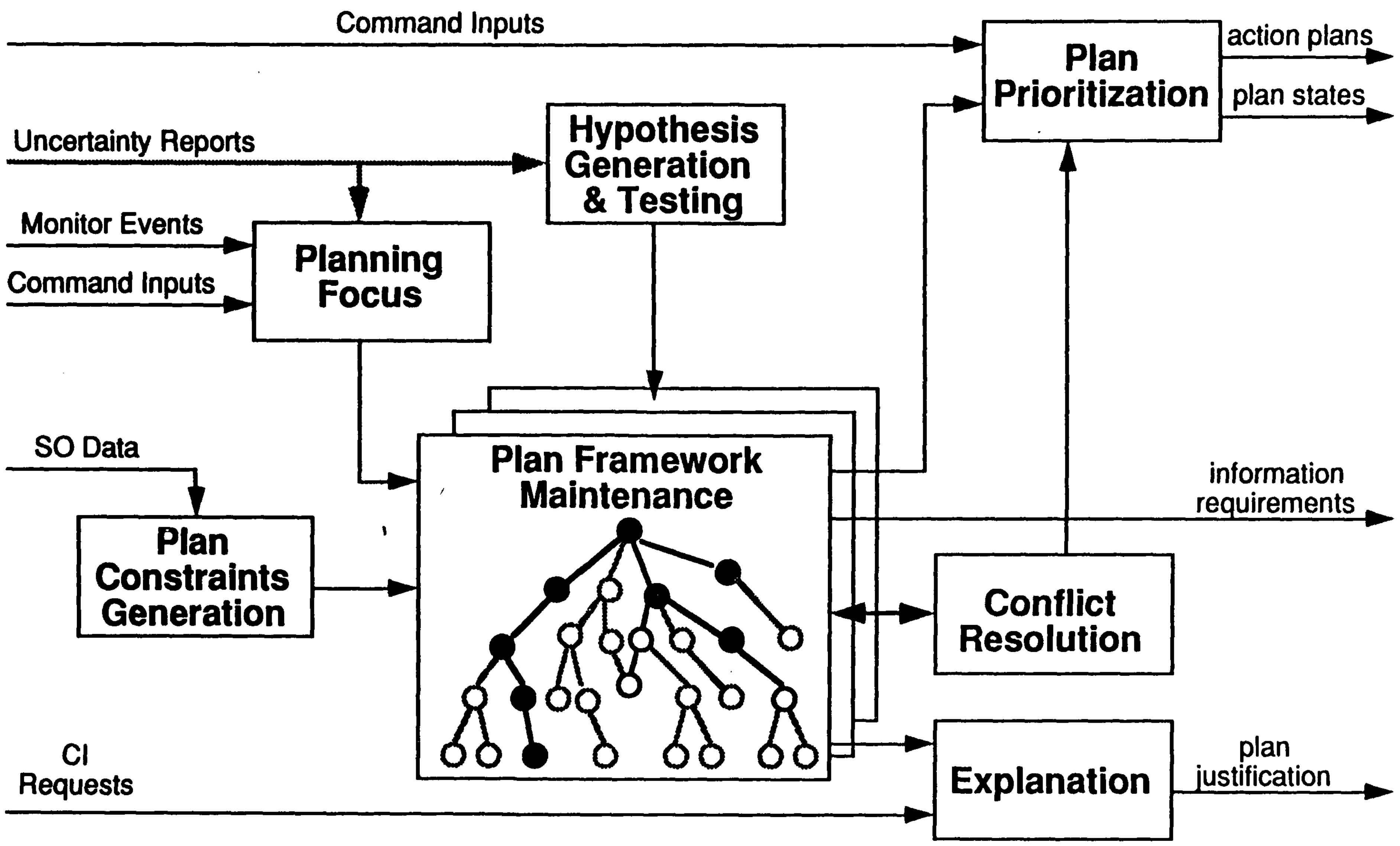


Figure 2-5. Tactics Planner Functional Flow

tems to share data, but also provides a unified context for all assessment and planning functions and an opportunity to resolve conflicts that arise among proposed plans. The System Manager was successfully adapted from the Mission Manager subsystem of LASC's Pilot's Associate program; and the Phase I SOAS program provided an opportunity to validate this important integration methodology. The functional flow of SM is shown in Figure 2-6, and further definition of the functionality and processing is presented in Section 4.4.

2.3 SYSTEM ARCHITECTURE AND INTERFACES

A major SOAS goal is to provide the submarine command with an enhanced situational awareness by analyzing sensor and submarine system information, distilling the large quantities of data into relevant information, and managing the presentation of that information to the command dectsion makers. From this information, corrective measures or alternative plans for achieving mission goals can be developed and presented to the decision makers for approval and execution. The Submarine Operational Automation System is not a "super-commander" or robotic submarine but rather an assistant to the decision makers who remain the final authority.

The Lockheed technical approach involves the division of the Submarine Operational Automation System subsystems into the general categories of "assessment" and "planning". The Situation Assessment (SA) subsystem monitors events external to the submarine; this assessment functions is focused by the current mission state, as represented by the active plans posted by the Tactics Planner subsystem.

Tactics Planner, on the other hand, performs planning activities exclusively. It offers the Commander multiple offensive and defensive options related to ASW operations. It also suggests plans for targeting analysis based on the validity of the target motion analysis and the measures of effectiveness for engagement planning computed by Situation Assessment.

The Command Interface subsystem involves both assessment and planning since it must monitor the Commander's activities and mission events in order to interpret command-level actions and configure the displays to optimize situational awareness and mission effectiveness. Monitoring command actions as as well the mission situation, the CI matches actions or specific commands to a model of the command-level plans and goals in order to determine, without explicit input, the intentions of the commander. This estimate of the command intent guides the behavior of both the assessment and planning functions.

Plan Coordination

Plan representation is a difficult concept to understand since the method of generation, the purpose for generation, and the methods for monitoring and explaining plans vary within the system. The TP subsystem creates very reactive plans, some of which continue to specialize or modify themselves even during execution. The CI, in contrast, maintains a static plan repre-

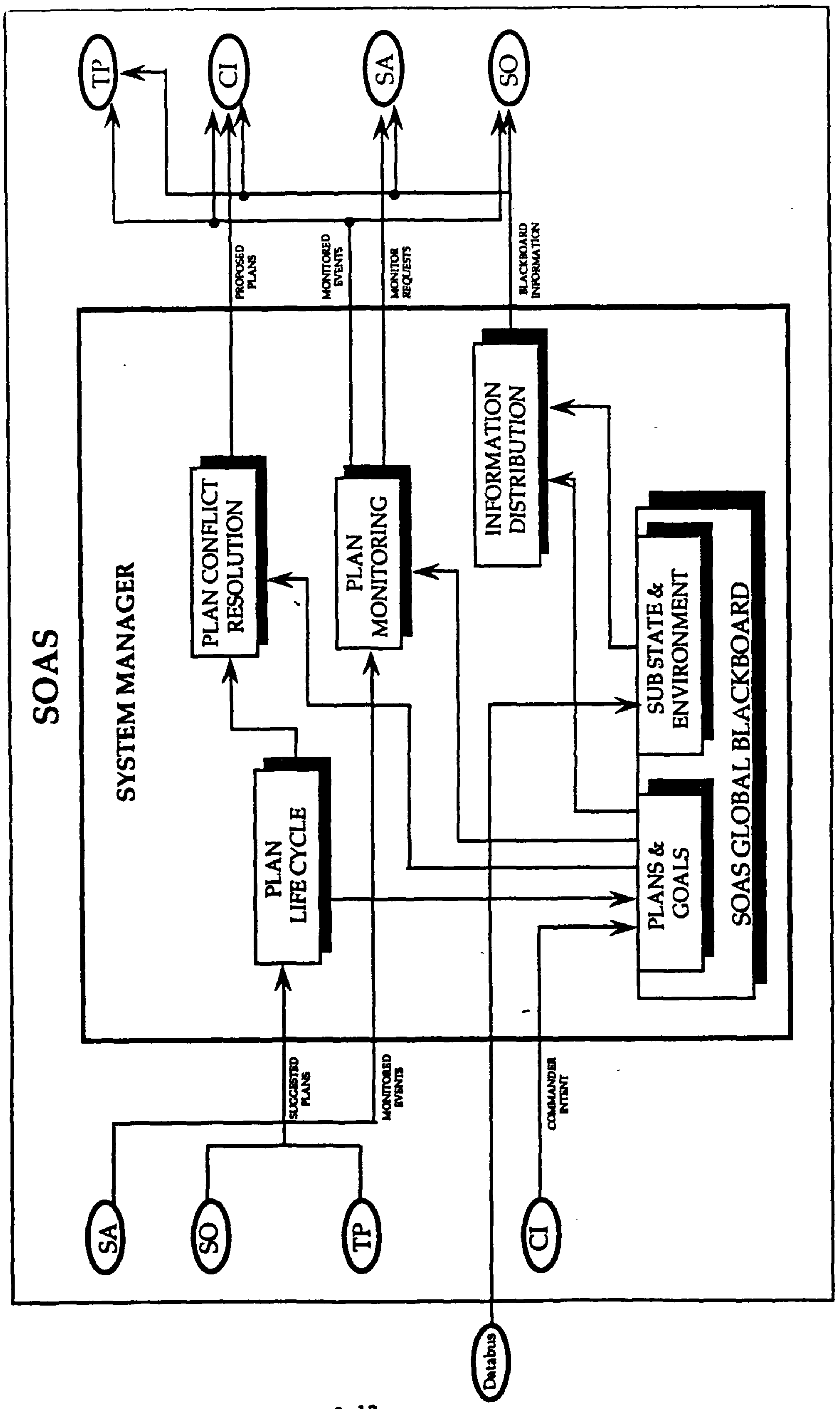


Figure 2-6. System Manager Functional Flow

sentation primarily to understand command actions, rather to optimize and recommend responses. In each case the plan representation is different, and yet both subsystems must exchange precise information about proposed and active plans.

The planning process must be controlled at the system level. The plans themselves must be organized from strategic to reactive, so that planning can begin at the more general level and details can be developed consistent with the top-level plans. In addition, the planning within each subsystem must be consistent with existing approved plans when possible, and conflicts must be identified and managed.

Plan Representation

In order to coordinate the plan generation and plan execution process across all SOAS subsystems, a plan and goal structure maintained in the System Manager has been developed as a means of coordinating the multiple planning activities within the system. This structure, represented as a Plan and Goal Graph, consists of planning elements hierarchically arranged to satisfy specific goals which are the children of higher level plan elements (Fig. 2-7).

It is important to realize that TP, SM, and CI have very different internal structures containing knowledge specific to their operations. The Plan and Goal Graph represents the common vocabulary with which these different subsystems communicate. The nodes of this graph are called "plan classes," copies of which become "plan images" when they are added to the Tactics Planner agenda.

Control of the Planning Process

Plan generation and execution in SOAS occur in three stages, as shown in Figure 2-8.

1. The tactical planning stage takes OOD and/or CO briefing information which initially constrains the next evolution and the overall mission. This process outputs a data structure called the tactical plan, which is a high-level representation of the expected mission events expressed as constraints on the route, time, resource allocations, etc. This data structure is the context within which more detailed planning must be done.

2. Plan selection takes current situation information, system information, and command intent and chooses the appropriate procedures for executing the tactical plan, including permitted ranges of key parameters. This is a set of typical responses to the environment, but not yet a set of actions to be performed.

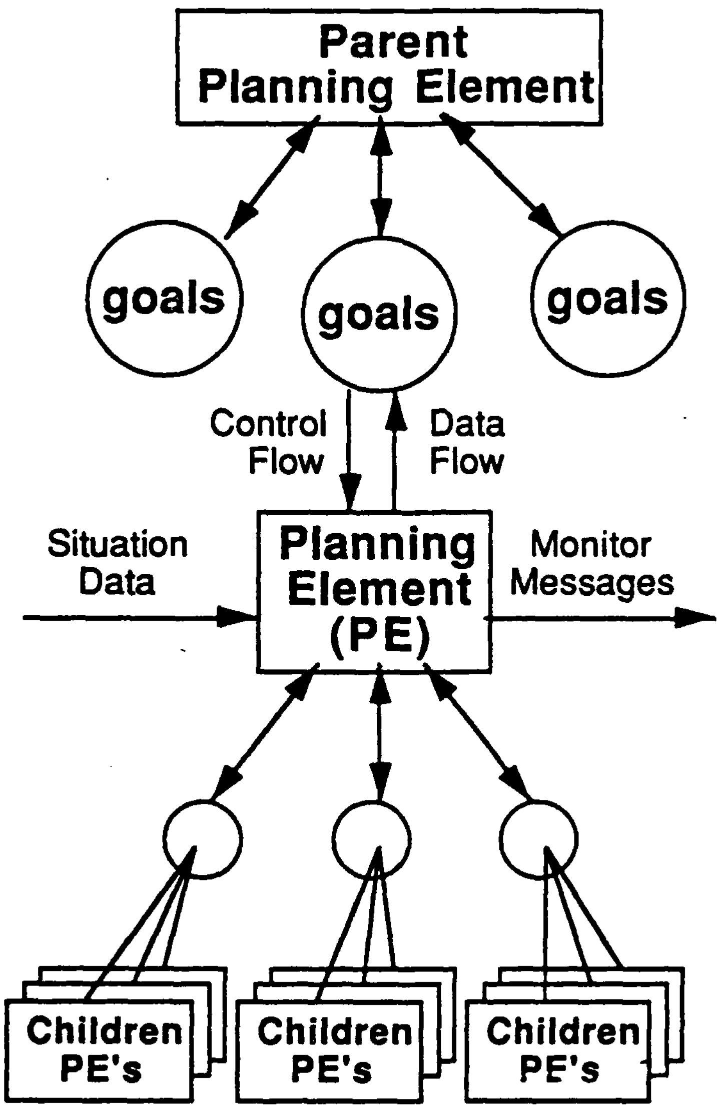


Figure 2-7. Plan and Goal Graph Example

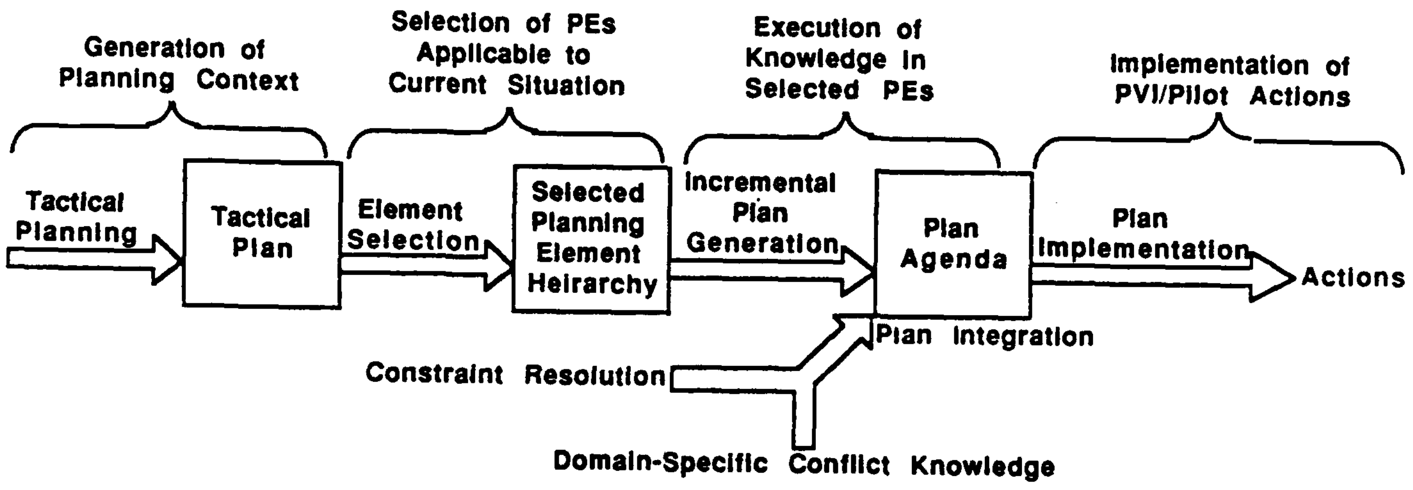

Generation of Planning Context
Selection of PEs Applicable to Current Situation
Execution of Knowledge In Selected PEs
Implementation of PVI/Pilot Actions
Tactical Planning
Tactical Plan
Element Selection
Selected Planning Element Heirarchy
Incremental Plan Generation
Plan Agenda
Plan Implementation
Actions
Plan Integration
Constraint Resolution
Domain-Specific Conflict Knowledge

Figure 2-8. Plan Generation and Plan Execution

3. Plan execution is the generation of specific actions. The selected procedures are "executed" by choosing specific values from the allowable ranges and producing an agenda of specific actions to be performed. The agenda of reactively-planned actions is carried out by the crew and/or CI subsystem, which operates only with the previously authorized permission of the command decision maker.

The plan element selection process begins at the highest level and proceeds to lower levels of detail as shown in Figure 2-8. At the lowest level, as specific agenda actions are considered, conflicts among actions appear and are resolved by specific knowledge at the lowest common parent node of the conflicting elements. Thus, the conflicts tend to migrate "upward" through the plan structure whereas planning constraints which resolve conflicts flow "downward."

The selection and specialization of plan elements should not be confused with the execution of plans. Plan elements are generated in response mission events and data, and may be posted simply as contingency for a likely event. A plan may be proposed for consideration by other subsystem, without ever being proposed to the command decision makers. Finally plans may be feasible and proposed, but never approved by the decision makers either by explicit rejection or implicit counteractions. All these examples serve to emphasize the distinction the between plan generation process and the execution and monitoring of suggested actions. The mechanism for relating the plan generation and execution is the planning agenda, illustrated in Table 2-1. This agenda is constantly maintained by the System Manager and serves as the input to CI for cueing the CO or decision makers about what actions are suggested to perform and when.

TABLE 2-1. System Manager Planning Agenda

ITEM	CHARACTER	EXAMPLE
Action Name	Text String	Hard Left Rudder
Value Range	Pair of Numbers	40 to 45 Degrees
Requesting Plan	Text String	Contact Analysis
Parent Goal	Text String	TMA Analysis
Priority	Number	17
Explanation	List of Goal Names and Parameters	Near-Object Classify Contact Determine Position Perform Mission

The plan/goal structure is essential to the SOAS development process. It provides a common system-level view that is shared by all subsystem developers. By identifying the various sources of plan information and the phases of planning and by translating this information into detailed, executable plans, it provides a common language of plans and goals (and their functionality in the system) that helps unify the specification of message traffic among the subsystems. It enables the integration of all planning activities by means of a single structural representation.

Developing a system-level representation of plans and goals has led to a better understanding of the problems of planning and plan interpretation in systems like SOAS. The panning process and the plan execution process, for example, pose different problems in terms of processing to be performed, data requirements, etc.

An essential distinction between the plan generators, such as Tactics, and the CI function, which explains the command actions in relation to existing or potential plans, has been clarified; and a mapping between the representation used in the TP and the CI plan and goal structures has been developed.

Important questions still remain, including how particular classes of plans can be included, and how the crew will participate in guiding or modifying the plan generation and execution process. However, the basic approach has been validated both in the Pilot's Associate program and in the Phase I SOAS program; and it provides an excellent framework for addressing these questions.

Approaches and Lab Architecture

The approches used by each of the demonstrated SOAS subsystems are listed in Figure 2-9.

The architecture used in Phase I to develop and demonstrate the SOAS subsystems is presented in Figure 2-10. It consisted of four Symbolics machines with four Silcon Graphics Iris workstations for displays. The predictive algorithmic routines of the Situation Assessment subsystem were located on a VAX 11/780 along with the simulation. Communication was performed with an Ethernet protocol.

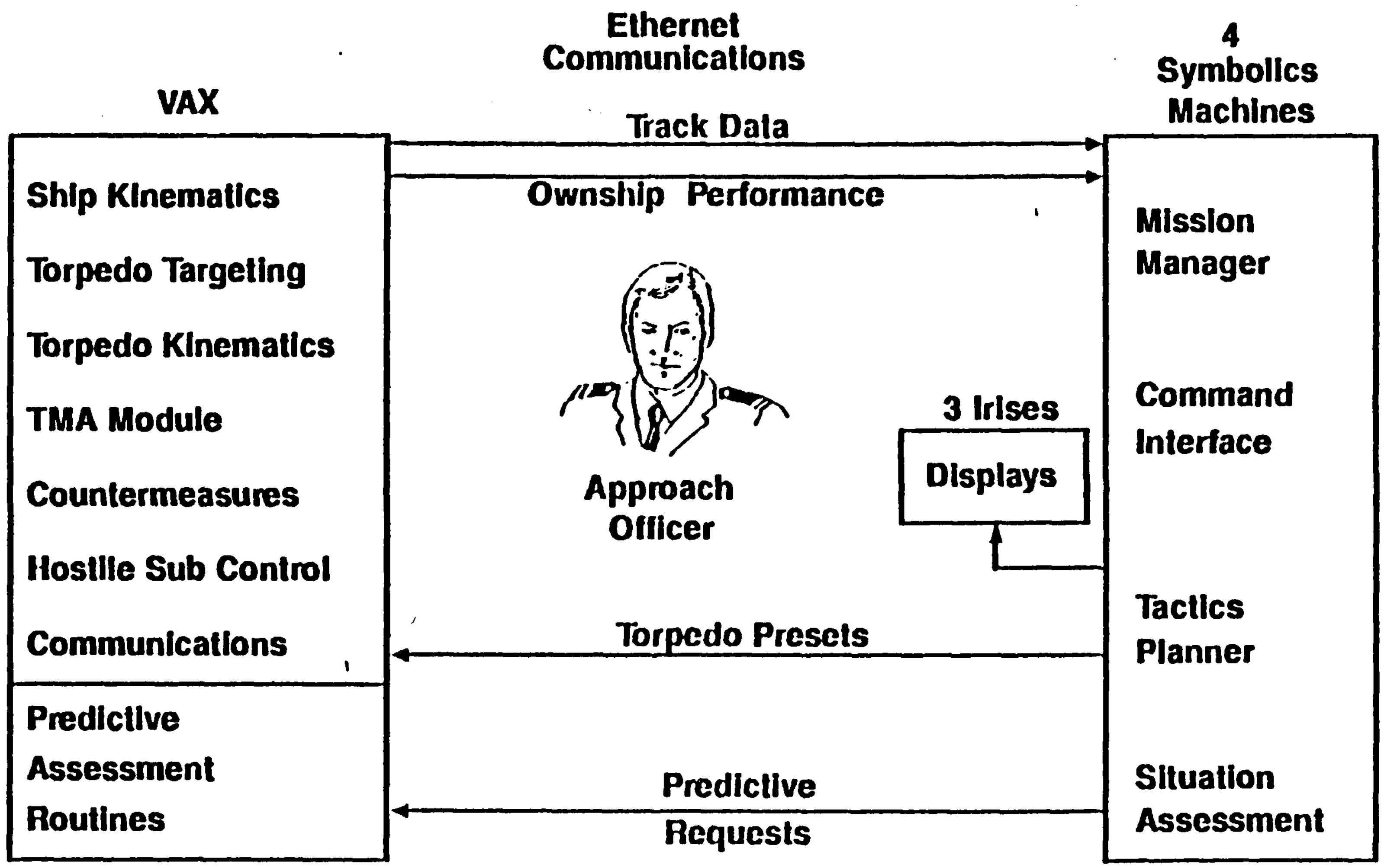


Figure 2-9. Phase I Laboratory Architecture

3.0 SOAS PHASE I DEVELOPMENT APPROACH

The process of defining, maintaining, and validating the interfaces by which the SOAS subsystems cooperate is at the heart of the Phase I development approach. The initial predictable need for a stable, configuration-managed Interface Control Document (ICD) was agreed at the offset. However, during integration of the prototypes described below, it became clear that a completely new type of document was required to draw together scattered pieces of information related to the plans in the system. The result was the Plan-Goal dictionary described in Section 3.3.

3.1 SUBSYSTEM INTERFACE CONTROL

Based on an Information Flow Diagram, it was possible to identify the information paths among the SOAS subsystems. For each path, it was necessary to specify the protocol by which the information was transmitted and the format of each message in that protocol. Figure 3-1 is an example drawn from that document illustrating the nature of the information captured in the ICD and maintained as a living document throughout the development process. Because this information was critical to the development of each subsystem, it was maintained under strict configuration control as outlined in the SOAS Software Development Plan (Rev A, 6 Dec 1989) and was available electronically for use by the subsystem developers over the Lockheed dial-up secure network.

3.2 SYSTEM PROTOTYPES

Since one of the primary technical issues on the SOAS program was integrated of the various expert subsystems, a strategy was developed and implemented whereby all the subsystems were brought together and integrated every three months. Even before this event, however, a trial integration process took place in the creation of the "scripted" System Manager (SMM). This was a version of the SM with additional rules to represent the behavior of each of the subsystems through a particular narrow slice of the scenario. Its original intent was to provide a "living" model of the message traffic in the SOAS which would validate the ICD and provide a model interface for the real subsystems to match. The first integration of the SMM achieved this objective by validating the first version of the ICD. Subsequent issues of the SMM were retained as substitutes for the real subsystems during partial system integrations.

3.3 PLAN-GOAL DICTIONARY

As the prototypes matured, they became more centered on the plans which were being proposed. The detailed mechanization of the evolution of a plan is described in Section 4.1.6. Since all the subsystems were communicating about plans and goals and each had different needs for information relating to those plans and goals, it became necessary to define the scope and relationships among the plans and goals. The first representation of this need was a Plan-Goal Graph, a sample of which is shown in Figure 3-2. This represents goals (a state of the world to be achieved) as ovals and plans strategies for achieving that state) as rectangles.

3.3.4.2.2 Periodic - rate TBD. Data of 9999 implies no available data.

```
(SA SONAR  <time> CONTACT_RPT
           <contact_id>
           <sensor_id>
           <range>
           <range_rate>
           <range_sd>
           <bearing>
           <bearing_rate>
           <bearing_sd>
           <signal_excess>
           <age>)
      <contact_id>       ::=  integer
      <sensor_id>        ::=  integer
      <range>            ::=  real (yds)
      <range_rate>       ::=  real (yds/min)
      <range_sd>         ::=  real (yds)
      <bearing>          ::=  real (deg)
      <bearing_rate>     ::=  real (deg/min)
      <bearing_sd>       ::=  real (deg)
      <signal_excess>    ::=  real (db)
      <age>              ::=  real (minutes)
```

Figure 3-1. Interface Control Document Example

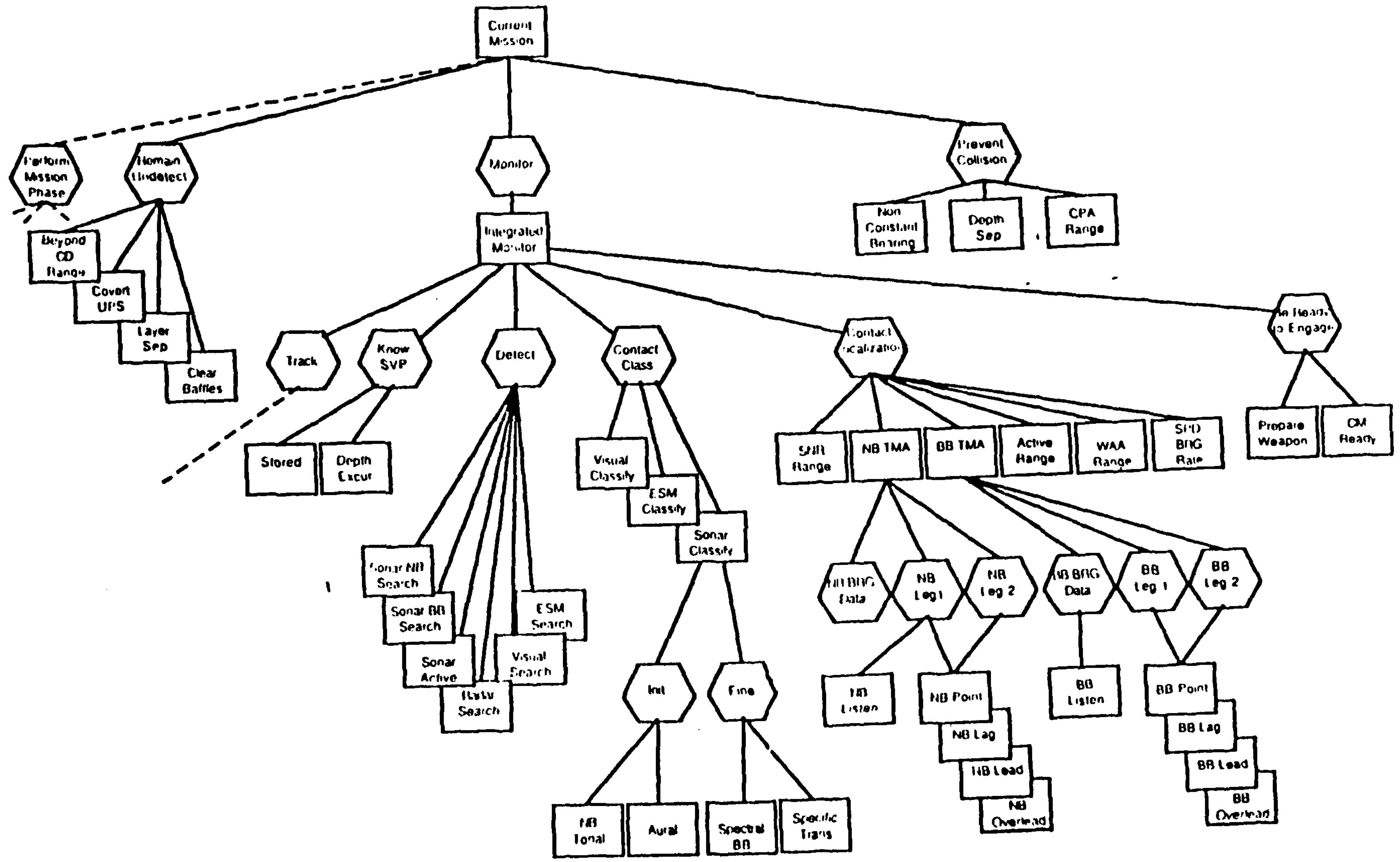


Figure 3-2. Plan-Goal Graph Example

A detailed algebra relating these concepts was formalized since plans contain enough knowledge to be able to divide their parent goal into subgoals and any goal could be satisfied by more than one plan. The graph, therefore, became a rather complex tree whose relationship were defined. Of all the plans which could satisfy a goal, any one of them shall be sufficient to do so. However, for the children goals of a plan, all child goals must succeed for the plan to succeed.

For plan generation, knowledge was required to select, specialize, and execute the set of plans necessary to solve a particular situation. For explanation of operator actions in this context, knowledge was required to distinguish the various plan-centered explanations of a particular action. For plan success or failure, knowledge of what parameters were important for plan termination and/or continued operation was needed. For plan display, information requirements for both suggesting and executing each plan had to be specified. All this knowledge about plans resides in software, in different places in the subsystems. However, it was clear that a central dictionary was required as a repository for all the scattered knowledge relating to a particular plan or goal. This became known as the Plan-Goal Dictionary, and the Plan-Goal Graph became a graphical index to that dictionary showing the explicit structure to the plans and goals. Figure 3-3 illustrates a sample from that dictionary.

```
(Goal-name             DO-TACTICS-PLAN
Source                 ""
PGG-Index              1-E-2
Module                 MM POSTS   TP SATISFIES
Function               IS SATISFIED BY TP MEANS PILOT
                       WANTS THE TP AS A RESOURCE
Prototype-Schedule
Characteristics        MULTIPLE  DYNAMIC
Exclusions             NONE
Legal-Parent-Plans     PERFORM-MISSION
Legal-Children-Plans   TOP-LEVEL-TACTICS
Parameters             NONE
)
```

Figure 3-3. Plan-Goal Dictionary Example

4.0 SOAS SUBSYSTEM DESCRIPTIONS

This section contains an overview of the four SOAS subsystems which were demonstrated during Phase I of the program: Command Interface (CI), Tactics Planner (TP), Situation Assessment (SA), and System Manager (SM). Presented for each subsystem is information about the following topics:

- Goals and objectives
- General methodology, including subsystem architecture and tools
- Module descriptions, explaining the technical results achieved during Phase I in terms of functionality implemented

4.1 COMMAND INTERFACE SUBSYSTEM

4.1.1 Overview of Command Interface

As its name implies, the Command Interface (CI) serves as the interface between key members of the attack party and the other systems in the submarine. In designing the CI, the philosophy adopted was that these key members should be "in charge." Thus, the CI was designed to support them in accomplishing their chosen goals. For the Phase I implementation, the Command Interface subsystem focused on the particular needs of the Commanding Officer (CO); thus, the following description is presented within the context of this one individual. However, in future phases, the CI architecture and functionality which have been developed are easily extensible to other personnel in the combat center/control room.

The primary function of the CI subsystem is interface management, which involves decoding inputs and encoding outputs, selecting information to be displayed, and configuring controls to meet the CO's current and short-term plans. Accomplishment of this primary function is supported by the secondary function of intent inference. Intent inference involves monitoring the CO's actions and inferring what he is trying to accomplish. The results of the intent inference function are vital to the CI and are also used outside the CI in other parts of SOAS.

As shown in Figure 4-1, these primary and secondary functions (i.e., interface management and intent inference) are accomplished in separate modules of the CI. Implementation of these functions has involved embedding a variety of models of the human, submarine, and situation into the CI modules. These modules will be discussed in detail in separate sections. The remainder of this Overview describes the Command Interface in terms of its inputs, outputs, and overall processing. It also explains the knowledge structures common throughout the subsystem.

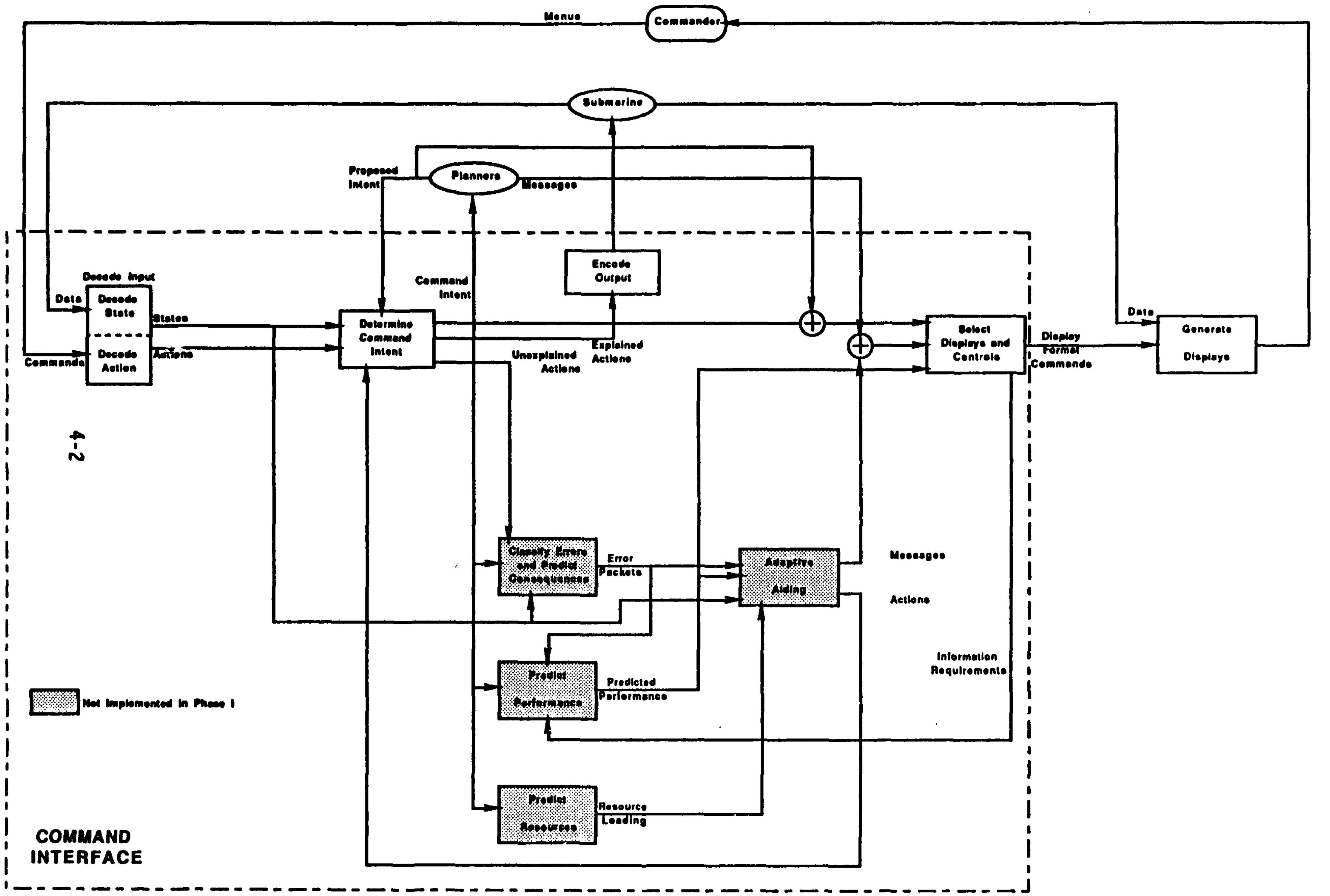


Figure 4-1. Command Interface Subsystem

Inputs and Outputs

In its role as intermediary, the CI interacts with three types of agents: other SOAS subsystems, the submarine simulation, and the Commanding Officer. The SOAS subsystems with which the CI interacts (i.e., Tactics Planner and Situation Assessment) are knowledge-based systems that propose plans and produce value-added assessments supplementing submarine and external state data. These "planners" provide proposed plans and text explanations of the proposed plans as input to the CI; the CI provides the CO's actual (inferred) intent as output to the planners. Information from the submarine (i.e., the submarine state) is obtained via sensors; the CI communicates with the submarine by issuing commands to actuators and sensors. The CO communicates with the CI by selecting options from menus; the CI presents information to the CO via visual displays.

For the Phase I demonstration, the Command Interface subsystem was presented in a stand-alone mode. Although all inputs and outputs were simulated, the overall processing within the subsystem was successfully executed.

Overall Processing

Processing in the CI is portrayed in Figure 4-1. In the following paragraphs, processing is discussed within the context of CI interaction with each of the three agents described (i.e., planners, submarine, and Commanding Officer).

First, consider the interaction between the CI and the planners. As noted, planners propose plans to the CI. These proposals are in the form of recommendations to be considered by the CO. For example, suppose Tactics Planner (TP) proposed to dive below the layer. This proposal would be passed to the intent inference function, where it would be stored and a meta-intent to consider the proposal would be created. The interface management function would then reevaluate the current displays in light of the proposed plan and modify them if necessary. For example, the interface manager might display a text message about the proposal and order the sound velocity profile (SVP) to be shown.

The Commanding Officer has the option to accept or reject a proposal. Further, acceptance or rejection can be communicated explicitly or implicitly. In the SVP example, explicit acceptance would indicate that the CO intended to dive below the layer; explicit rejection would represent explicit CO disapproval of the proposal to dive below the layer. Implicit acceptance or rejection is inferred by the intent inference function, based on the CO's actions. For example, if the CO issued a command to dive to a depth that was below the layer, then the plan would be implicitly accepted. Changes to the CO model, if any, are communicated to the planners and the rest of the CI.

Next, consider CI interaction with the submarine. Submarine data are decoded by the information management function into the common state representation used by all CI modules. The submarine state information does not directly prompt processing by CI modules. However, the intent inference function may run synchronously, and changes in submarine state are processed by this module at the next invocation. Commands from the CO are represented within the CI as actions in a common action format; the interface management function encodes these actions into appropriate commands for use by the CO or systems of the submarine.

Finally, consider CI interaction with the CO. As noted, the CO may issue a command by menu activation. The input decoding function translates the command into a common action format used internally by all CI modules. Next, the CO's command, represented as an action, is passed to the intent inference function for interpretation. If the action can be interpreted by the intent inference function as consistent with current or feasible CO intent, the action is considered explained and is passed through output encoding to the CO or systems for execution. If the intent inference module interprets the explained action as representing a change in the CO's intentions, the change in intent is communicated to the planners and the rest of the CI.

Once the CO's tasks are known, the interface management function determines how information is to be displayed. Displays and controls are selected on the basis of current and proposed CO intentions. For example, an intention to prevent counterdetection by diving below the layer would be supported by displaying the SVP and providing a means to issue commands (e.g., a menu). Once the order to dive below the layer has been issued, the SVP information might not be needed any longer and the displays and controls would be changed to support new intentions.

In addition to supporting CO, selection of displays and controls considers display constraints and human information processing limitations. Information is prioritized based on importance and deadline, and the more important information is selected for display first. Once the appropriate displays and controls have been identified, display selection commands are passed to the display generator, which creates the actual displays viewed by the CO.

In summary, the CI serves as the intermediary between the user, the submarine, and other portions of SOAS. The philosophy guiding the design of the CI was that the Commanding Officer is in charge of the system. The primary function performed by the CI in support of the CO is interface management. This function is supported, in turn, by the secondary function of intent inference in which models of the CO are embedded. The primary and secondary functions are performed by highly interdependent modules within the CI.

Common Knowledge Structures and Software

Several knowledge structures used throughout the CI are described here to avoid repetition. These include blackboards, goals, plans, scripts, actions, and states. Goals, plans, and scripts represent CO intent.

A slot-filler construction is used to represent all entities throughout the CI. Each slot-filler structure is of a single type (e.g., goal, state, or action) and consists of one or more slots (or attributes) and a filler (or value) for each slot. For example, the structure used to describe the goal

Remain Undetected would be of type "goal." Its slots would include "name" and "actor." The "name" slot would contain the filler remain-undetected and the "actor" slot might contain the filler "CO," which would indicate that the CO was pursuing this goal.

Slot-filler structures are generalized to store more complex forms of information. For generality, a filler may also be another structure or list of structures all of the same type. In addition to storing a literal value such as "CO" or remain-undetected, a filler may also be a named variable such as "*actor-var*." Such structures are called patterns and are used in pattern matching and instantiation, as described below.

Pattern matching is the comparison of two structures, one of which is data and the other a pattern. Some of the pattern's slots contain variables as fillers. The pattern-matching process compares the two structures to determine if they match. Two structures match if they are of the same type and all slots in the pattern match all slots in the data. Two slots match if they contain the same filler, or the pattern contains a variable as a filler which, in turn, has a value equal to the data filler or has no value assigned. If the pattern variable has no assigned value, the matching causes the variable to receive a binding, which may then serve to constrain future pattern matching. The pattern matching returns bindings if matching is successful.

The instantiation process modifies a pattern structure by replacing one or more named variables with values. Instantiation need not be complete, i.e., only some variables need to be given values; the remaining slots with variable fillers will not be changed.

CI rule-based functions use a common inference engine named Chainer, which supports both forward and backward chaining. Forward chaining works from known facts to conclusions (i.e., deductive reasoning). Backward chaining works from conclusions to initial causes or facts (i.e., hypothetical reasoning). The direction of inference and the conflict resolution strategy can be controlled by either the Chainer package or the calling function.

Blackboards

All information used by the CI modules resides on either the library blackboard or the regular blackboard. The library blackboard contains the original entities in their uninstantiated form and information that does not change during execution (e.g., knowledge bases). The regular blackboard is dynamic in that it holds the current state of the CI. It contains instances of goals, plans, scripts, states, etc. Throughout the remainder of this section, references to the CI blackboard should be interpreted as referring to the regular blackboard unless otherwise specified. Blackboards within the CI should not be confused with the blackboards of the other planners.

Goals

A goal is an object that describes a desired or expected state of the CO, submarine, or external world. It serves as the parent of a set of plans, each of which is a method to achieve the state represented by the goal.

Goals have a number of slot-filler pairs. The name is a symbol used to retrieve and identify the goal. The planner is the agent pursuing the goal. The attributes represent the preferences of an agent for a particular goal relative to other goals. The end state represents the states of the CO, submarine, or world that satisfy the goal. If the states are described by variables, there may also be a conditions clause that compares the variables to particular contexts. The goal also has a start time, expected duration, and its actual satisfaction time. A goal's recurrence requirements are represented as single or continuous. A goal may be satisfied by a single occurrence of its desired end state, or it may by its nature require continuous satisfaction.

Goals may serve as substeps of a plan. The relationship of a set of goals to its parent plan is that all sub-goals must be achieved for the parent plan to succeed. Plans are related to their parent goals in a logical OR fashion. That is, one or more of the plans must be completed in order to satisfy the parent goal.

Plans

Plans represent non-structured methods for pursuing goals. A plan is an object that describes the salient parameters of a method but allows for easy distinction between it and other related but different activities.

Plans have a number of slot-filler pairs. The name is a symbolic reference for retrieval and identification of the plan type. A separate, unique symbolic tag is provided to identify specific instances of the plan. The planner is the agent who is performing the plan.

The attributes are a set of characteristics that represent the cost of a plan. These cost attributes interact with the value system of the planner to determine the relative preference of the plan compared to other plans that might satisfy a given parent goal. A plan also contains temporal information such as the starting time, the expected duration, and actual duration of the plan.

There are slot-filler pairs that specialize the plan for each instance of its occurrence. An example might be the "track" slot, which will have a variable filler that allows the plan to be specialized to a particular instance involving a particular track.

Scripts

Scripts are sequences of activity with particular ordering constraints. A script is a particular method for executing a plan. A checklist or procedure may be implemented as a script. A given plan may have no scripts associated with it, or one or more scripts that could be used to fulfill the plan. When a plan is activated, one of its scripts (if it has any) is also activated. In order to choose a script when several possibilities exist, a script has a set of attributes that represent its preference to he CO.

The basic building block of a script is the event, which is made up of four distinct parts: action, state, constraint function, and side-effect. Actions are commands that can be sent to the submarine. States are the internal representation of the submarine state vector. The constraint function places conditions on both the action and the state to limit the context to the particular needs of the event. The side-effect function allows any inter-process communication about the event that may be needed. An event is said to occur if either its action or state is observed and its constraining conditions are true. When an event occurs, it is marked as having occurred and its side-effect function is evaluated.

The script body is the sequence of events that make up the execution of the method. The body is organized into a set of segments, each of which can contain an arbitrary number of events. The events inside a segment can occur in any order. Two kinds of segments are defined: blocks, in which each event should occur only once; and cycles, in which each event can occur an arbitrary number of times. The final block of the body is the event that signifies the normal ending of the script.

Actions

An action is a command that can be sent to the action systems of the submarine. Actions result in submarine state changes, which are then sent back by the submarine to the CI. Typical actions include setting torpedo presets and commanding a new depth or speed.

State Vectors

Each system in the submarine is represented as a state vector. Each state vector has an "actor" slot containing the system name, and a slot for each attribute of the system that the CI uses. The state vector also has a time-stamp to show when it was last updated. The state vectors may contain both quantitative and qualitative data.

4.1.2 Command Intent Module

The intent inference function is accomplished in the Command Intent module. This module supports the integration of the CO's plans and goals with those of the other planners in the SOAS system. The integrated set of plans and goals forms the basis for communication between the CO and SOAS and for providing the CO with a managed display and control environment in the command station.

Command Intent Module Functions

The functions of the Command Intent module are as follows:

1. Explain CO actions in terms of a script (i.e., checklist or procedure) previously known to be active.

2. Explain CO actions in terms of plans previously known to be active.

3. Explain CO actions via hypothetical reasoning about feasible new plans and goals, and provide these new plans and goals to the other SOAS planning functions.

4. Identify abandoned scripts, plans, and goals.

5. Identify inconsistent actions.

Command Intent Module Software Design

The Command Intent module is designed as three major components, all coded in Common Lisp: the executive, the script-based reasoner, and the plan-based reasoner. The components form a general-purpose intent understanding system called Operator Plan Analysis Logic (OPAL). The OPAL system uses functions from the set of general-purpose utilities that support the CI modules to perform its low-level data access, matching, and rule-based inference processes.

The OPAL functions have been implemented in two components. Functions 1 and 4 are implemented in the script-based reasoner, and functions 2 and 3 are implemented in the plan-based reasoner. Function 5 is not directly implemented but is a by-product of functions 1, 2, and 3.

The executive component provides access to the central CI blackboard data structure, thus buffering the other two modules from the details of the CI interface to the other SOAS subsystems, as well as from the other CI modules. The executive component calls the other two components, depending on the status of data on the CI blackboard. At the conclusion of an execution cycle, the executive updates the CI blackboard with the results. Figure 4-2 shows a functional flow diagram for OPAL.

Script-Based Reasoning

The script-based reasoning process uses a set of pre-stored procedural representations called scripts to find explanations for each action of the CO. Scripts have three major structural parts:

1. Script top, which contains identification and timing data

2. Script termination clause, which lists events causing the script to be considered deliberately stopped by the CO

3. Script body, containing the procedural steps to be performed. The script body represents procedural data as an ordered set of segments, each containing one or more events.

The structural components of scripts are described more fully in Table 4-1. The CI blackboard contains the set of scripts known to be active at any given moment.

The script-based reasoning process is a linear, iterative process that scans the active scripts on the blackboard and marks each associated event

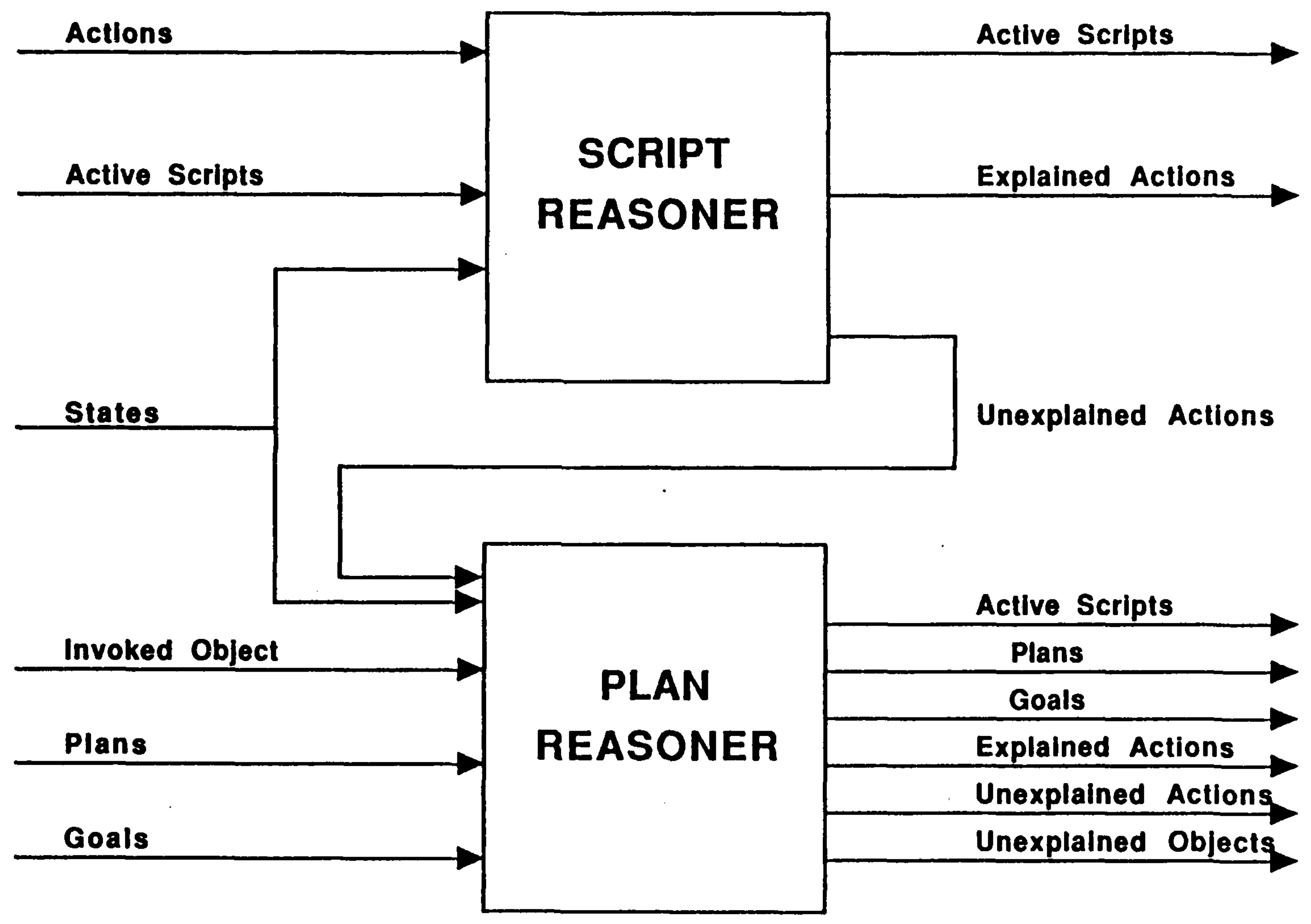


Figure 4-2. Intent Inference

TABLE 4-1. Script Components

Slot Name	Description
Action	A representation of a CO party input that contains variables for the prior and commanded state of the submarine or system, but does not indicate how the action was entered.
State	A representation of the value of a particular submarine or system parameter. Normally, actions cause a change in state.
Bindings	A list of variables and their values. Bindings are used to specialize actions to particular values of systems or states.
Condition	A boolean function that can compare bindings or values of states directly from the blackboard.
Side-Effect	A function that alters the data environment of the reasoning process without affecting the direct inference that is made.
Event	The smallest unit in the execution of a script. The event is the fundamental building block of scripts used in both the terminations and body. An event is made of an action, a state, a condition, and a side-effect. It can be marked when it has been done, and may be optional or required.
Segment	A group of events which can be marked in any order. Two types of segments exist, Blocks and Cycles. In Blocks, each event can only happen once. Cycles allow indefinite repetition.
Terminates	A special segment that holds the events which cause the script to be no longer active.
Body	A list of segments which must be done in the exact order in which they are listed. The last segment provides for the deactivation of the script on its normal completion.
Script Top	A set of parameters that describe the script in general and identify each particular instance of the script. The top includes name, ID, and timing data.

as accomplished as it is observed to occur. Three stages of scanning are performed on each script.

1. First, the script top is scanned to determine if its allowable execution time has passed. Scripts which have not been completed prior to their execution time are marked as abandoned scripts.

2. Second, each script is scanned for terminating events in its termination clause. If a terminating event has occurred, the script is immediately marked as terminated and removed from the set of active scripts. If the terminating event was satisfied by a CO action, the action is not explained by the terminating event and further search for explanation must be performed.

3. Finally, the body of the script is scanned. The scanning process is constrained to consider events in only one active segment of each script at a time. This ordering constraint allows events inside a single segment to be performed in any order, but requires that all events in a segment be completed prior to the activation of the next segment. If an input action satisfies an event in the body, then it has been explained by OPAL as an event that was expected on the basis of the CO's intent to continue performing a known active procedure represented by the script. The body of the script is updated to show the current status of the script, including completion of any segments. Branching to other scripts is performed as a side-effect of events in a script whenever appropriate.

After all active scripts have been scanned, the script-based reasoner returns control to the OPAL executive, which updates the CI blackboard with the current active scripts, including the effects of script abandonments, script terminations, and script updates. This update occurs each OPAL processing cycle even if there was no CO input, because of the need to scan scripts for events that may be satisfied by the current state of the submarine. If there was no input action or if the action was explained by a script event, OPAL returns control to the CI. Otherwise, the plan-based reasoning component is called to continue the attempt to explain the CO's action.

Plan-Based Reasoning

The plan-based reasoning component searches the SOAS plan and goal graph in a bottom-up fashion, trying to find a path in the graph that connects the input action to some known plan or goal. The plan and goal graph is represented as a rule base, in which each link between a parent object and its set of children is a rule. The rule format conforms to that required by the Chainer utility component. The parent object is found in the IF clause of the rule, and each of its children is expressed as a premise in the THEN clause of the rule.

The relationship between a particular child and its possible parents is constrained by two processes: a binding list and a set of constraint functions.

- **The binding list** specifies values for variables that exist in both the child and the parent. These bound values allow the general pattern of an object, whether action, plan, or goal, to be particularized to a specific instance of the general pattern. For example, the plan to go to a waypoint named X only satisfies the goal of being at a place Y if X and Y are bound to the same waypoint identifier.

- **The constraint functions** allow very general comparisons of the values of specific bound variables and values from the internal or external environment of the submarine as represented on the CI blackboard.

Each rule is also capable of executing side-effects, which modify the data environment of OPAL but do not directly affect parent-child relationships. Script activation is an example of such a side-effect. When a plan is inferred as part of the attempt to explain an action and the plan was not already known to be active, the direct action of the rule will put the plan on the active plan list. If the plan is supported by a script, the side-effect of the rule will activate the script so that future CO inputs that represent the continuation of the plan can be explained by the script-based reasoning component of OPAL. Other side-effect functions are the removal of conflicting plans and goals and the creation of new variable bindings.

The rule base in the plan-based reasoner is stored as a hash table that permits direct access to the rules that could apply to the current reasoning state. The rules are used in both the forward and backward directions during the search for an explanation of an input action, plan, or goal. The search process is recursive and consists of two major steps: prediction and find-parent.

- **In the prediction step,** a rule that links the current object (action, plan, or goal) to a known active plan or goal is sought. Rules are fired backward from the current object to find a possible parent pattern. If a rule fires backward, the rule is then tried forward from the known set of plans and goals. If it also fires forward, this implies there is a plan or goal in the known lists that is an acceptable parent of the current object. The known lists then can be said to predict that the current object is consistent with the CO's intentions.

- **The find-parent step** fires rules backward from the current object to find its parent. The newly found parent will serve as the current object in the next cycle of recursion. If no new parent can be found for the current object because either bindings or constraints prevent any rules from firing, then the search for an explanation has reached a dead end. The search backs up by returning to the prior level of recursion.

At each level of recursion, a local environment containing the results of all side-effects, such as conflicting plan removals and activated scripts, is maintained. When the prediction step is satisfied, this local environment is saved and returned to the OPAL executive for posting to the CI blackboard. The input object has been explained by the plan-based reasoner, so it is posted to the appropriate blackboard slot depending on the type of the input object. If the search returns to the input level of recursion without satisfying the prediction step, then the input has no explanation. Objects without explanation are posted to the CI blackboard for further analysis.

The plan-based reasoner is used by OPAL to explain both CO actions and plans proposed by the other SOAS planners. The OPAL executive calls the plan-based reasoner only if there are objects on the blackboard to be explained. If both a CO action and a proposed plan from another planner are on the blackboard, the plan-based reasoner may be called twice in a single OPAL cycle. An essential product of OPAL's processing is the updated plan, goal, and script lists formed during the generation of an explanation for actions and plans. These lists are supplied to all other modules of the CI as vital inputs to their processes. They are also supplied to the SOAS planning functions to indicate the CO's intentions in terms that map directly into their internal reasoning processes.

The plan and goal graph that forms the basis for OPAL's reasoning includes explicit plans for accepting and rejecting the proposed plans of the SOAS planners. Because of this, the CO may express his intentions directly by explicit acceptance or rejection of the proposed plan, or he may express his intention indirectly by merely taking action. The explanation of the action by OPAL will confirm either that he is acting in a way that is consistent with the proposed plan or that he has chosen a different plan that still achieves the known goals and satisfies the constraints.

Command Intent Module Knowledge Representation and Acquisition

The knowledge required by the Command Intent module consists of scripts and rules. The operational procedures for the simulated submarine and for CO interaction with SOAS constitute the major source of script knowledge. Specific knowledge required includes the set of all possible actions at the command station, the effects of each action on the simulated submarine systems and SOAS, and the constraints on each action in terms of submarine and system states that are represented in the simulation. Representations of these actions are used to create the script events.

The knowledge used to construct OPAL's rule base is extracted from the plan and goal graph. Each plan and goal in the graph is represented as a pattern containing its variable and fixed parameters. For each plan and goal, the sets of all plans and goals which conflict must also be identified. For plans performed by clearly established procedures or checklists, scripts are identified. All other plans must be decomposed either into subgoals or directly into actions from the set of all possible actions (as discussed earlier). Finally, the constraints that limit the feasibility of parent-child relationships between the actions, plans, and goals must be identified. All these relationships are used to build rules for the rule-based reasoning component.

The number of rules and the complexity of their interaction are a direct reflection of the granularity of the representation of intentions. During the course of Phase I, three different plan and goal decompositions were performed, producing representations of intentions that were progressively finer grained and broader in scope. The current design of the Intent Module does not depend on any particular granularity. It should be noted, however, that the attainable execution speed of the module is directly affected by the number of plans and goals used to build the rule structure.

4.1.3 Interface Manager Module

The Interface Manager (IM) is responsible for managing the displays and programmable controls to meet the CO's information needs. IM also decodes incoming messages and encodes outgoing messages for the CI. Even though IM operates autonomously, the CO may override its selections at any time.

Interface Manager Module Functions

The Interface Manager has three primary functions:

1. Select displays and controls.
2. Decode inputs to the CI.
3. Encode outputs from the CI.

Selecting displays and controls (function 1) is quite complex and is accomplished via the following secondary functions:

1.a Determine logical devices to be evaluated.
1.b Determine information to be displayed.
1.c Allocate displays to logical devices while maximizing information displayed.
1.d Select text messages to be displayed to the attack party.

Interface Manager Module Software Design

The ways in which the above functions are accomplished in IM are discussed in the following paragraphs. To avoid excessive nesting of headings, the secondary functions associated with display and control selection are listed at the same level as input decoding and output encoding.

Determining Logical Devices to Be Evaluated

Determination of logical devices is illustrated in Figure 4-3. IM determines whether or not a physical device should be considered for display replacement based on the time since the last display change. This function serves to limit IM's use of processing resources and to permit CO-chosen displays to remain visible. IM then orders physical devices according to the focus data. This data is used to cause the most important information to be displayed on the devices where the CO most frequently looks. The focus data is currently determined by knowledge engineering rather than by measurement.

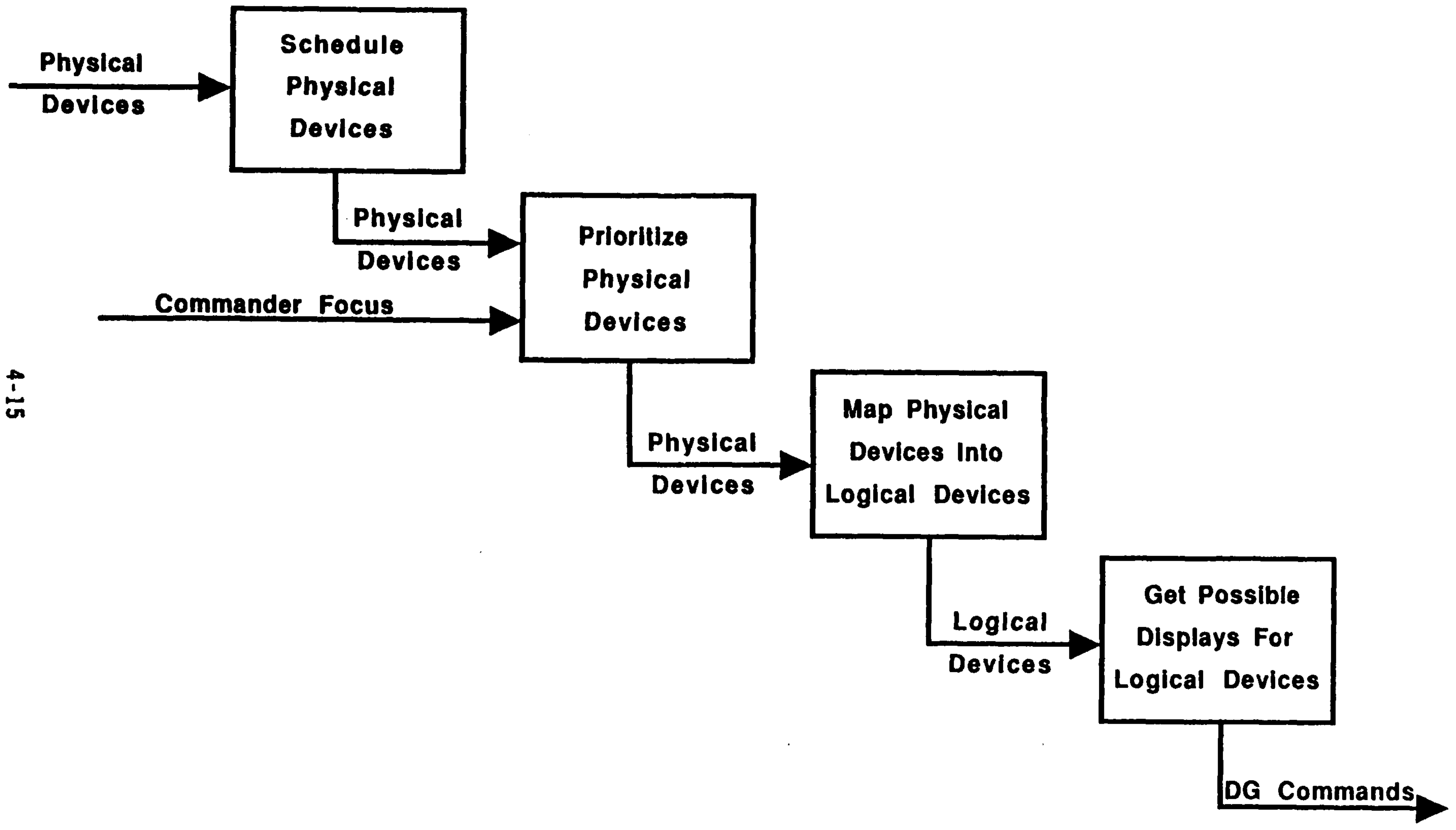

Physical Devices
Schedule Physical Devices
Physical Devices
Commander Focus
Prioritize Physical Devices
Physical Devices
Map Physical Devices Into Logical Devices
Logical Devices
Get Possible Displays For Logical Devices
DG Commands

Figure 4-3. Select Logical Devices to Get Displays

Each physical device may be divided into one or more logical devices. This is necessary when two important types of information compete for the same physical device. Choosing one or the other may be less desirable than showing both in smaller areas. Division of display space is currently constant.

Each logical device has a set of displays that may appear on it. Several logical devices may be capable of showing the same display, but it will be chosen only once. In order to show a menu, a display, and a text message on the same physical device, it must be divided into three logical devices. The potential displays are currently fixed for each logical device.

Determining Information that Should be Displayed

This function, which is graphically displayed in Figure 4-4, involves integrating active and proposed plans, integrating the information requirements of plans, and removing low importance information requirements.

Integration of active and proposed plans involves removing mutually exclusive plans from the active plan list. This is valuable because it gives the CO a clearer picture what the proposed plans are. The intent inference function is invoked to merge the proposed plans with the active plans. The intent inference processing would have to be done on a hypothetical copy of the plan and goal graph because the proposed plans are not being accepted by IM. This is already part of the intent inference function in that proposed plans accepted by the CO must be integrated into the plan and goal graph.

Every plan has an associated list of information elements. This list represents the information the CO needs to execute the plan. For example, when coming to periscope depth, the CO needs to know the range and course of all contacts, the layer depth, ownship trim status, ownship depth, speed and course, and other information. The integration of these information requirements merges the requirements of all plans into a single structure. The result is termed the integrated information requirements.

Once the integrated information requirements have been identified, low importance integrated information requirements are deleted if necessary.

Allocating Displays to Logical Devices

Allocation of displays to logical devices is illustrated in Figure 4-5. This function is accomplished iteratively by choosing the display that covers the maximum of integrated information requirements by the current set of plans. After a display is chosen, the information it is capable of displaying is removed from the integrated information requirements. These reduced integrated information requirements are used in subsequent display selection decisions.

Each display is penalized according to the integrated information requirements that it cannot display. Then with a straightforward application of the Common Lisp sort function, the least penalized display is moved to the beginning of the list of feasible displays.

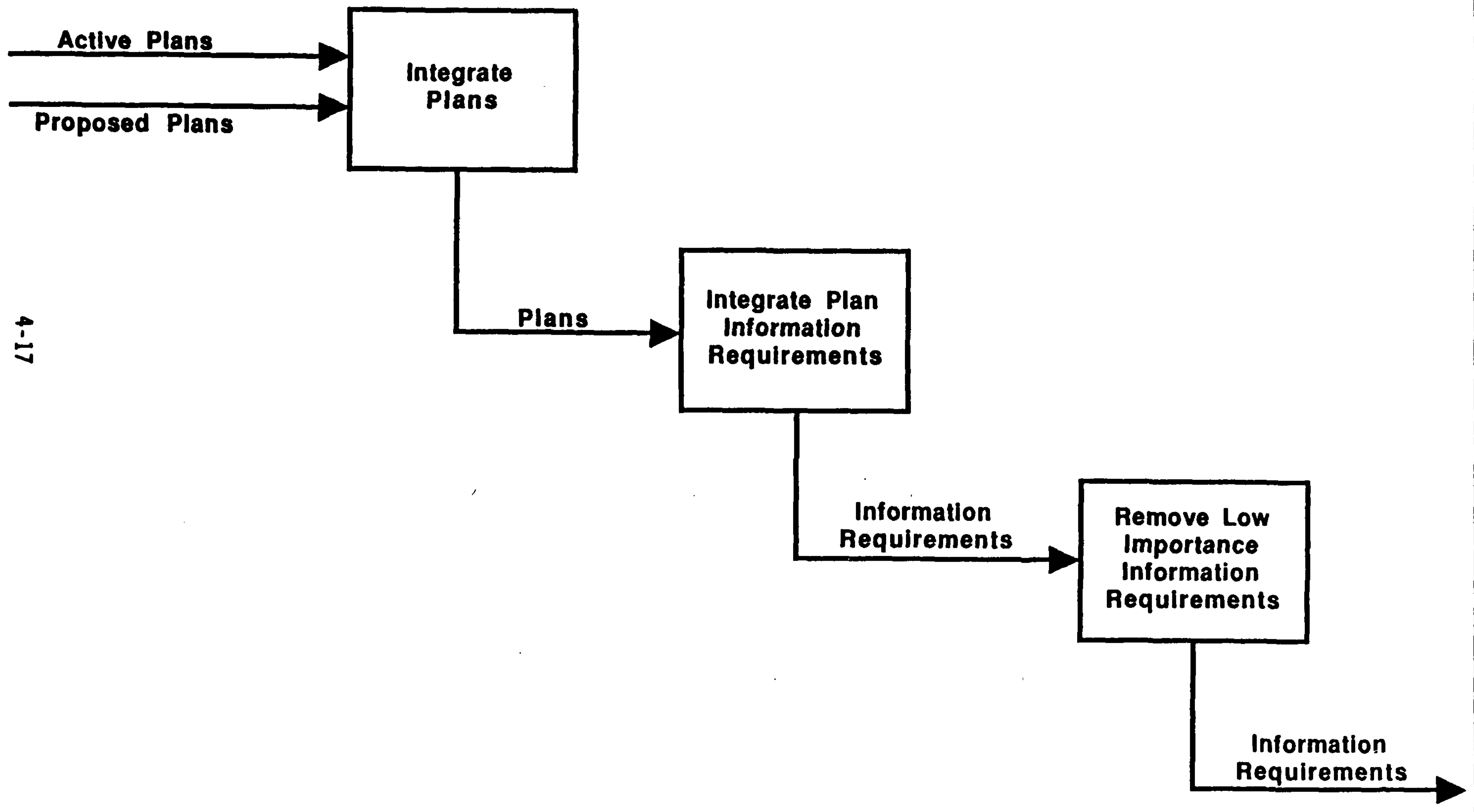


Figure 4-4. Select Information Requirements to Satisfy

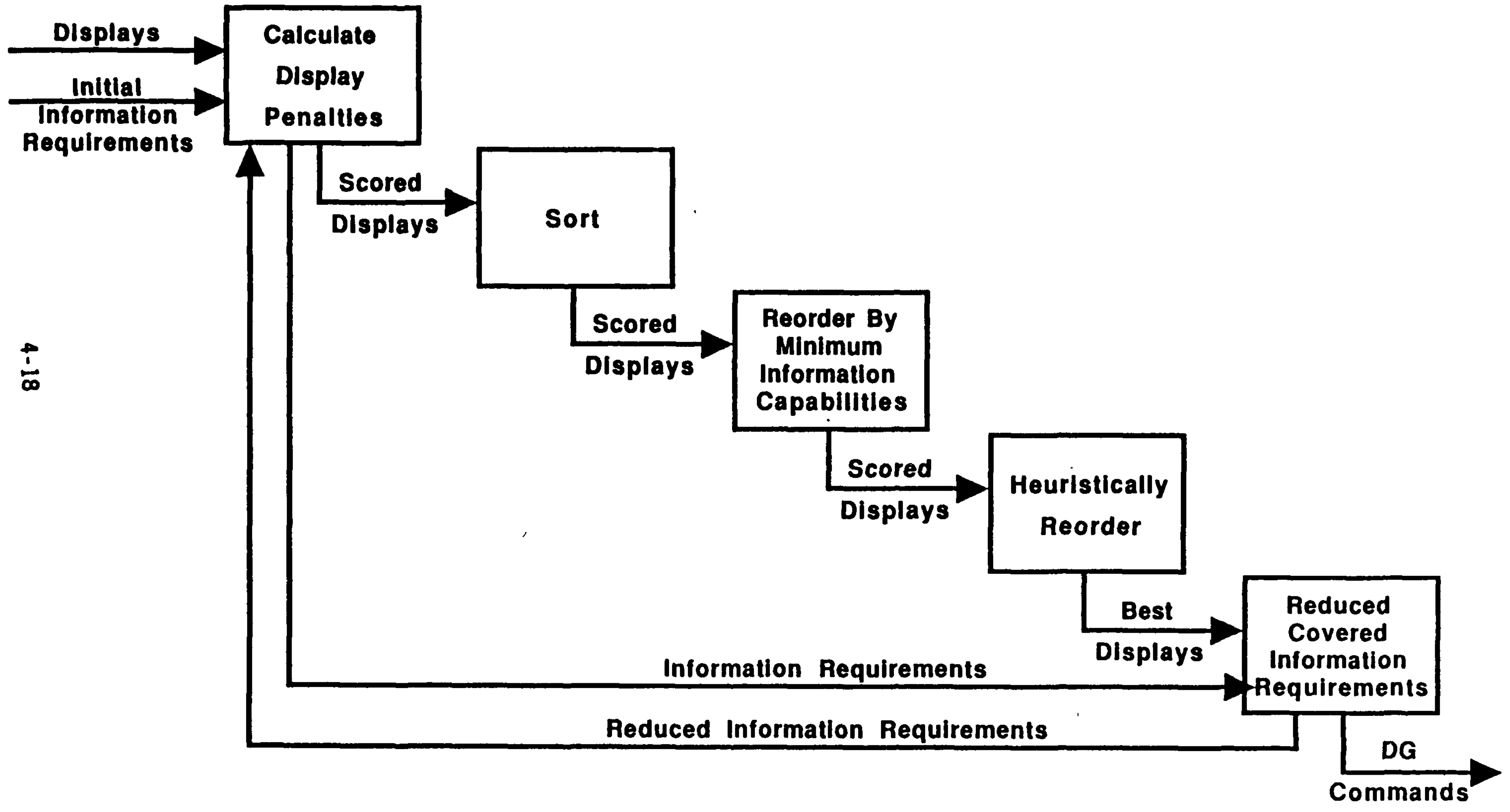

Displays
Initial Information Requirements
Calculate Display Penalties
Scored Displays
Sort
Scored Displays
Reorder By Minimum Information Capabilities
Scored Displays
Heuristically Reorder
Best Displays
Reduced Covered Information Requirements
Information Requirements
Reduced Information Requirements
DG Commands

Figure 4-5. Select Displays to Meet Information Requirements

Frequently, a number of displays receive the same numerical score. If this is occurs, then all the displays that receive the best score are sorted again based on the amount of information each is capable of displaying. The display with the least information is preferred, since it is the least cluttered.

Display assignment heuristics take into account situational aspects not considered by the scoring process. These heuristics can cause the selection of a display other than the one with the least penalty. An example is that when displays score close to the same value, and one of them is currently shown, no change should be made to the display format if it has been shown for less than a specified number of seconds. The heuristics also allow the CO's direct request to override selections made by IM.

After a display has been selected, its information capabilities are removed from the integrated information requirements. These requirements are used in subsequent passes of the allocation of displays to logical devices. At the completion of processing, the unfilled information requirements are an output of the selection process.

Selecting Text Messages to be Displayed to the CO

Selectng text messages is illustrated in Figure 4-6. Planners (e.g., Tactics Planner) produce text messages to tell the CO some information or to explain a proposed plan. Text messages appear on logical devices that can show only text. he process by which text messages are selected is described below.

Ordinarily, the number of text messages sent to the CI is expected to be smaller than the number of logical displays for showing text messages. Thus, although most of the logic in this function is unnecessary in the routine case, it is used in the worst case.

The initial specification of a text message provides a time at which the message need no longer be displayed. When a message's time limit is exceeded, it is removed from the logical device. In addition, logical devices are freed for subsequent text message display (as described below). If a text message is not displayed before its deadline, the planner that prepared it is advised of the failure. The text message is then discarded.

IM must also determine the importance of a text message. Each text message contains a topic and a list of plans to which it is relevant. IM finds the message importance by taking the maximum importance of the topic to all of the active plans in the list. Having considered both importance and display deadlines, IM sorts text messages into a priority order. A rule base is used to process the deadline information so that text messages that could wait are moved to the end of the queue.

It is important to determine the appropriateness of a text message for each available logical device. Two knowledge bases are used to accomplish this. One knowledge base correlates text message importance with the rate at which the CO samples a device. Devices that the CO samples more frequently

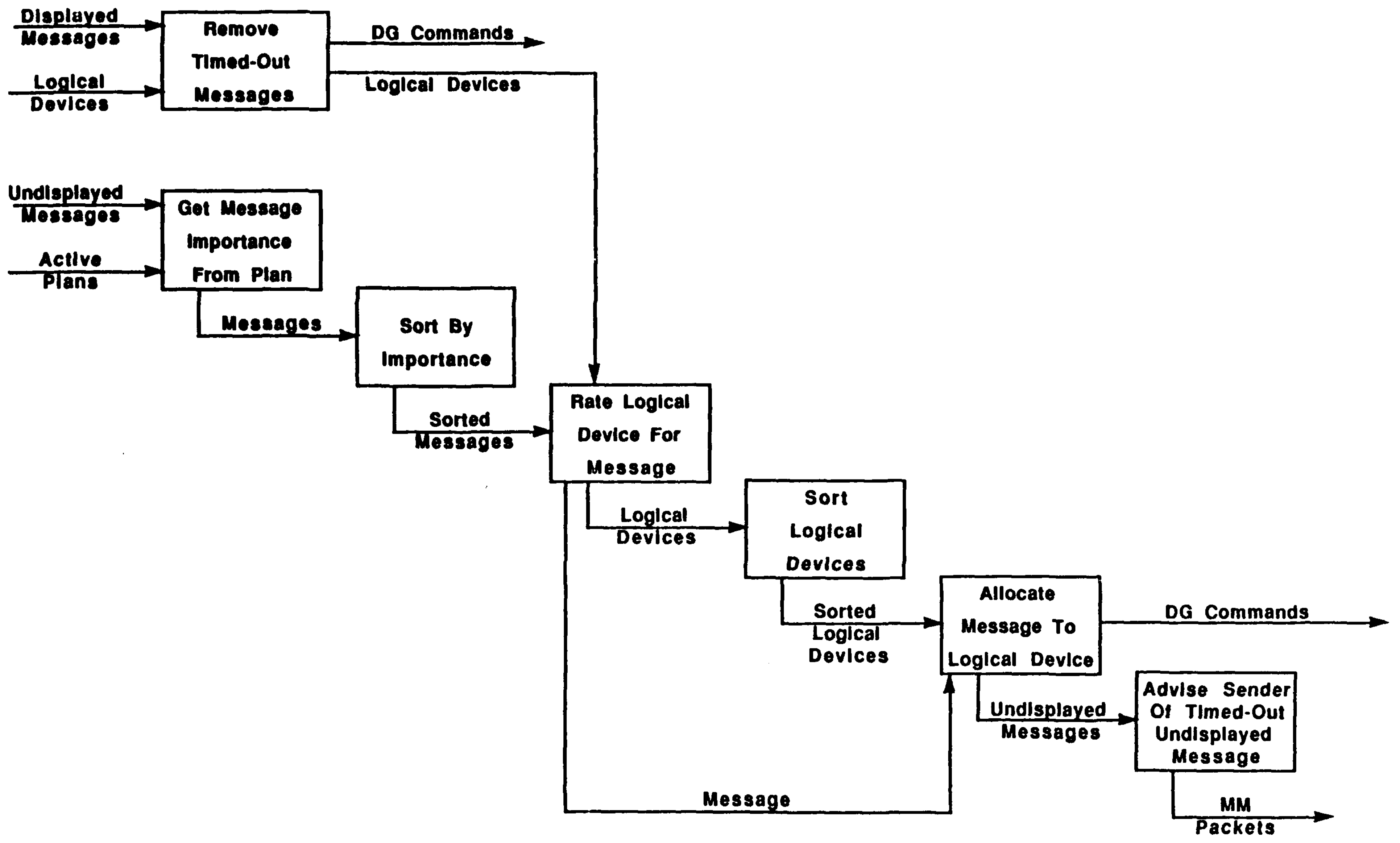


Figure 4-6. Text Message Selection

are rated higher. The other knowledge base helps enforce the rule that a text message which appears near a display should refer to the same topic. For example, a text message about the torpedo tube status would be shown near the tube status display since they both share the same topic.

The most important text message is placed on the best logical device. A knowledge base is used to select the color of the text message according to its importance and topic.

Input Decoding

Inputs from the CO are converted to the common internal representation by table lookup. Submarine state changes are also decoded into the internal representation by table lookup. Inputs from other subsystems are decoded and placed on the CI blackboard.

Output Encoding

Once a set of displays has been selected, the specifications are sent as commands to the display generator (DG). Once a display device has been configured, it dynamically presents changes in the represented information, as directed by the control logic of the particular displays and without explicit control from IM. Touch panels are controlled as displays so that the meaning of touching a particular part of the panel is conveyed to the CO by the graphics image overlaid on the panel. Internally represented actions are encoded into a form appropriate for commanding the submarine and sent, as commands, to the appropriate system for execution. Outputs to other subsystems are encoded and placed on the CI blackboard for output processes to handle.

Interface Manager Module Knowledge Representation and Acquisition

The knowledge acquisition process for plan information requirements is done as follows. The expert is given a plan to rate. He chooses the topics that are relevant to the plan from a predetermined list. Then the expert rates the relative importance of each topic to the plan. Finally, the expert rates the desired resolution, scope, bandwidth, and control bandwidth of each topic. The process is repeated for every plan.

Regarding knowledge representation, the interface manager uses a number of objects internally to represent the console configuration and information shown.

Physical Device

A physical device is either a display device or a control. The available displays are the overhead, left, center, and right CRTs. The available controls are the touch-sensitive menu pads on the center CRT. The internal representation of a physical device describes its location within the console, the decoding table that is used to decode its inputs, and the time and actor (i.e., CO or IM) of the last display selection.

Logical Device

A logical device is a virtual display device which corresponds to all or part of a physical display device. For example, currently each CRT is divided into three virtual devices. A logical device is created to display information about a single topic (e.g., ownship state) or set of logically related topics (e.g., all sensor systems).

The internal representation of a logical device includes its physical device parent, its screen boundaries, its neighboring logical devices, the actor and time of its last display change, and the names of the feasible displays that may be shown on it.

Display

A display is a dynamic picture that can be shown on a virtual device. The display is represented twice with the SOAS system. Within the display generator, the display is a computer program that can draw dynamic pictures on the CRTs. The data that are displayed come directly from the simulation or from messages that originate in various planning modules. Within IM, a display is represented in terms of its ability to display information. IM is responsible for choosing which displays are visible. The display generator is responsible for drawing the display on the screen.

The internal representation of a display contains the name of the logical device on which the display is located and the information that it is capable of displaying (i.e., a list of information elements).

Information Element

An information element is an abstract entity that represents both a plan's need for information and a display's capability for showing information. An information element contains a "topic" that describes what the information is about. Some example topics are ownship depth, tube 1 status, and contact state. When an information requirement is represented, an importance rating is included that indicates the importance of the information relative to the success of the plan. Importance is rated on a scale from one to ten, where one represents "nice to know" and ten represents "essential to plan success."

Other slots in an information element include input bandwidth, control bandwidth, resolution, and range.

- **The input bandwidth** is the CO's sampling frequency for information on the display.
- **The control bandwidth**, by analogy, is a rating of how frequently the CO could activate the control.
- **Resolution** and **range** refer to the precision and breadth of the information, respectively.

It should be noted that all of these dimensions refer to the CO's capabilities rather than those of the electronics. For example, bandwidth represents the frequency with which the CO can extract information from the display, not the frequency with which the display is updated. The resolution is the precision with which the CO can extract information, not the precision of the electronics itself.

Text Message

A text message contains a text string for the CO to read. It can be produced either within or outside the CI. IM decides how and when to display the text message without examining the text itself. The text message is described by a topic and a set of plans to which the text message is relevant. This is sufficient for IM to determine how important a text message is to the CO. The text message also describes the deadline by which time the CO should read the text.

Focus Data

The fraction of attention the CO allocates to each display is represented by focus data. It contains the name of a physical device and a numeric value that represents the attention paid to the physical device relative to other physical devices.

4.2 TACTICS PLANNER SUBSYSTEM

4.2.1 Overview of Tactics Planner

The two goals of the Tactics Planner subsystem for Phase I were to conceive and complete a design of the subsystem for Phases II and III, and to demonstrate the correctness and feasibility of the design. This was done in three steps. First, the basic operational and technical requirements were defined. From this, the second step consisted of choosing the design, tool, and basic approach which would be used to provide these requirements, including deciding which requirements could be provided immediately and which would have to wait for further development in later phases. The third step was the production of a Phase I software demonstration which provided a proof-of-concept of the suggested solution.

The next three sections give a summary of each step of this process. Section 4.2.2 presents the basic operation and technical requirements. Section 4.2.3 describes the Kadet planning tool, which has been chosen as the basis for the Tactics Planner subsystem. Section 4.2.4 summarizes the software demonstration, showing the basic functionality and the advanced features that were added during Phase I development.

4.2.2 Tactics Planner Requirements

The first stage of Phase I was to complete a requirements analysis for the Tactics Planner subsystem. The requirements were based on the defined goals of tactics planning within the SOAS system.

The purpose of the Tactics Planner subsystem is to support planning during actual situations. During tactical situaticns, the submarine commander is continually required to reason about and decide between competing options. Many of these decisions are based on parameters or information that is uncertain or incomplete, making the decision more difficult. The Tactics Planner subsystem will 'reason along' with the Commander.

To support the Commanding Officer in this fashion, Tactics Planner must reason intelligently about the submarine tactics domain. This domain features a number of potentially interacting subproblems which can overlap in several dimensions, including time (both planning time and plan execution time), computational resource demand and utilization, and implementation resource demand and utilization. Further, as stated above, the domain is riddled with uncertainty, which adds the requirement of doing cost/benefit analysis of uncertainty resolution. All these subproblems are in addition to the basic planning problem of handling multiple, possibly conflicting, plan options in a timely manner.

The remainder of this section briefly lists the technical functional requirements that are necessary to provide these functions to the command decision maker. (These functions are described in more detail in the System Development Plan.)

The Tactics Planner is designed to provide decision support to the CO. This involves technical functionality along two lines. The first is the ability to reason in the tactical domain to generate tactical options, planning optimizations, information trade-offs, and other information that is required. The second is to communicate these results to the commander in a timely manner. This second line of functionality is coupled to the functionality of the Command Interface subsystem.

The first technical challenge is to plan in the tactical domain. The following functionality is required:

- Dynamically generate, maintain, optimize and execute plans
- Provide focus of attention
- Reason about constraints, including constraints related to mission objectives, system status, current threats, and commander intent
- Resolve conflicts among competing or cooperating plans
- Reason with multiple plans and provide plan prioritization
- Reason with uncertain and incomplete information, including reasoning about uncertainty resolution
- Conduct real-time processing on real data
- Reason about time-critical limitations

The second technical challenge is to communicate the results of planning to the commander. The following functionality is necessary:

- The ability to reason about plan prioritization.
- The ability to use advanced explanation facilities.
- The ability to handle hypothesis testing.

4.2.3 The Kadet Tactics Planner

The purpose of the SOAS Tactics Planner is to provide support for the command decision process. The functional requirements necessary for the Tactics Planner to achieve this goal were outlined in the last section. All of these abilities must be present in any SOAS tactical operational scenario. The system must also have the ability to smoothly integrate further operational scenarios with no loss of capability.

The Tactics Planner subsystem will fulfill these functional requirements by using the Kadet planning system. Kadet is a reactive planner which utilizes skeletal planning elements to model operational situations. Kadet is ideal for solving the SOAS tactical planning problem for many reasons. The Kadet architecture was designed along with the Plan and Goal Graph methodology of knowledge acquisition. This allows for a straightforward transition of domain-specific knowledge into rules and rule-sets to drive Kadet. The remainder of this section describes the basic Kadet architecture.

The Kadet planning framework is based on a design and development methodology that uses predefined skeletal plans. Skeletal plans, developed as "generic" solution structures for carefully designed and bounded goals, are captured through knowledge engineering sessions. They can then be dynamically and continuously specialized to the changing situation based on a-priori collections of knowledge designed for this purpose. This knowledge is provided through the use of a versatile, yet understandable, rule system. Skeletal plans are organized into a solution hierarchy of planning elements that have been optimized to provide a framework for both plan generation and plan understanding and explanation.

The concept of skeletal plans has been augmented in the Kadet framework to include the following features:

1. Localized blackboard-oriented plan elements providing non-monotonic reasoning capabilities.
2. Script-based plan elements to support strategic planning.
3. Rule-based, multiple instantiations of plan classes to pursue parallel (cooperating or competing) responses to the planning context.
4. User-defined modularity to support intelligent resource allocation based on the planning context.

Much of the structure for tactical planning is common to all planning elements. This basic planning paradigm, identified in the course of this development effort, has driven the development of the Kadet planning tool. This generalized planning architecture is best understood by viewing a generic planning element as shown in Figure 4-7.

This plan has several components. First, each plan has a name. Each plan also contains a script, a predecessor/successor list of steps which must be performed in order to carry out this plan. The script often has only a single step, or it may have a set of steps operating in parallel. More complex scripts feature some steps in sequence, with others in parallel. Leaf node plans have empty scripts. Each script step represents the need to select a child plan to perform some activity; so each script step has a set of legal plans attached to it. Following the script is a set of initial assertions, facts asserted on the local blackboard which is contained within each planning element. The plan has a cardinality associated with it, which enables the plan to be instantiated only once (i.e., only one such plan can exist at any time) or as needed (such as a torpedo evasion plan, which may have a separate copy in action for each contacted torpedo); this slot contains a rule for determining this cardinality. The parent goal and child goals slots contain system goal names which identify this plan's position in the Plan and Goal Graph. Finally, the plan element includes several domain-specific rulesets: the transition, decomposition, execution, specialization, selection, and initialization rulesets. These rulesets are used to handle the two steps of planning: building the planning hierarchy and executing the planning hierarchy. These are termed the selection phase and execution phase, respectively.

- **The Selection Phase,** illustrated in Figure 4-8, is the first part of planning. When a selection message is sent from a parent, it means that this plan has been selected as a possible option for achieving some goal. (The initialization ruleset, which is not shown in the figure, contains basic control rules used in every planning element to handle specialization.) The specialization ruleset is used to specialize this planning element to the current situation. Specialized parameters are stored on the local blackboard. The selection ruleset is used to determine how well this planning element 'fits' the goal it has been selected to achieve, and the cost/benefit of continuing this line of planning. This information is sent to the parent. Depending on the selection ruleset, select messages may be sent to child planning elements to satisfy the subgoals. This selection phase is used to control building the planning tree, to focus attention on relevant planning elements, and to prune the planning process based on the current situation.

- **The Execution Phase,** shown in Figure 4-9, is the next part of planning. When an execute plan is received from a parent, it means that this planning element should attempt to execute to solve the goal. This does not mean that this planning element has been chosen for real-world execution, but only that it has been chosen

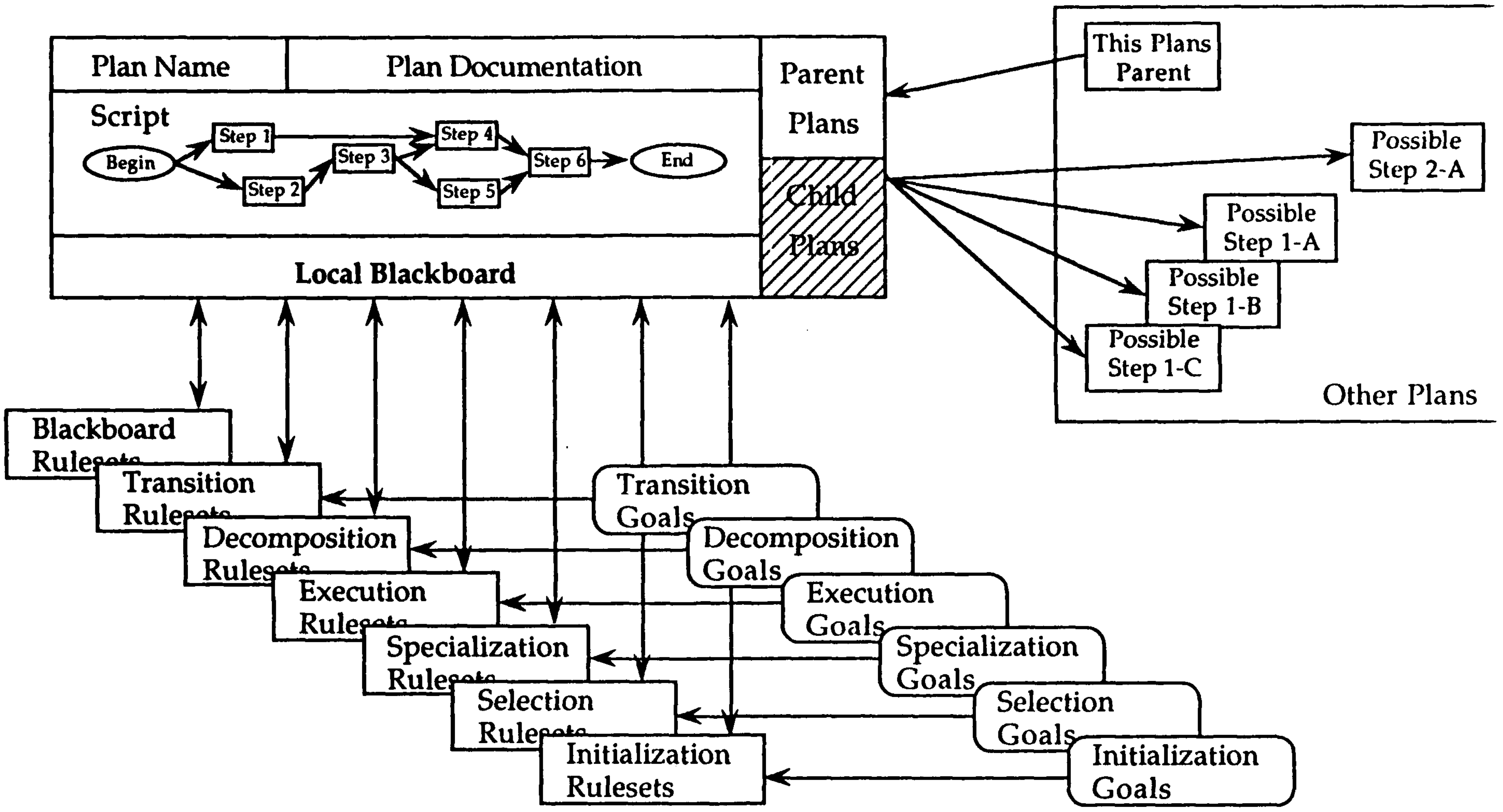


Figure 4-7. Kadet Planning Element

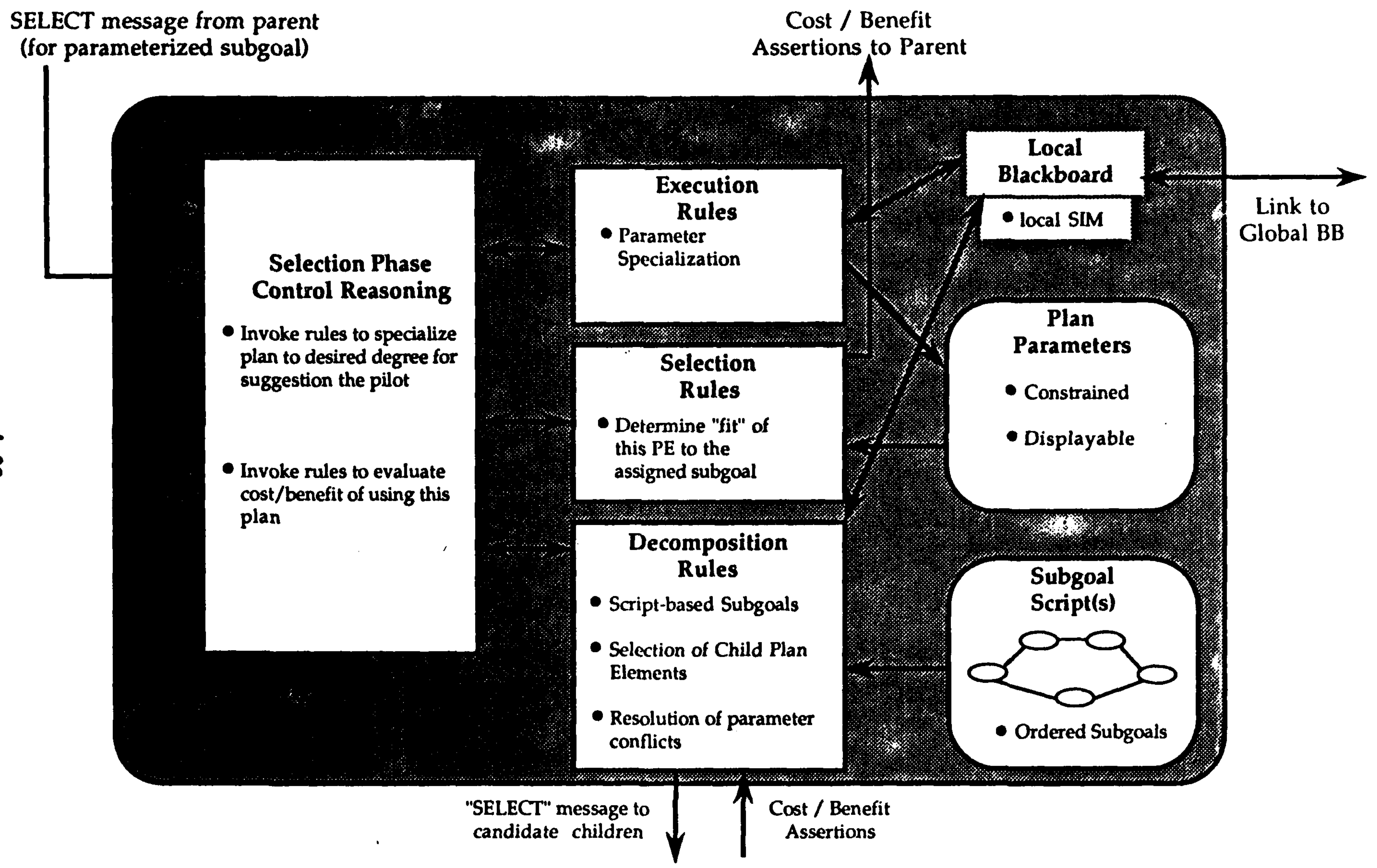

SELECT message from parent (for parameterized subgoal)
Cost / Benefit Assertions to Parent
Selection Phase Control Reasoning
● Invoke rules to specialize plan to desired degree for suggestion the pilot
● Invoke rules to evaluate cost/benefit of using this plan
Execution Rules
● Parameter Specialization
Selection Rules
● Determine "fit" of this PE to the assigned subgoal
Decomposition Rules
● Script-based Subgoals
● Selection of Child Plan Elements
● Resolution of parameter conflicts
Local Blackboard
● local SIM
Link to Global BB
Plan Parameters
● Constrained
● Displayable
Subgoal Script(s)
● Ordered Subgoals
"SELECT" message to candidate children
Cost / Benefit Assertions

Figure 4-8. The Selection Phase - Building the Planning Hierarchy

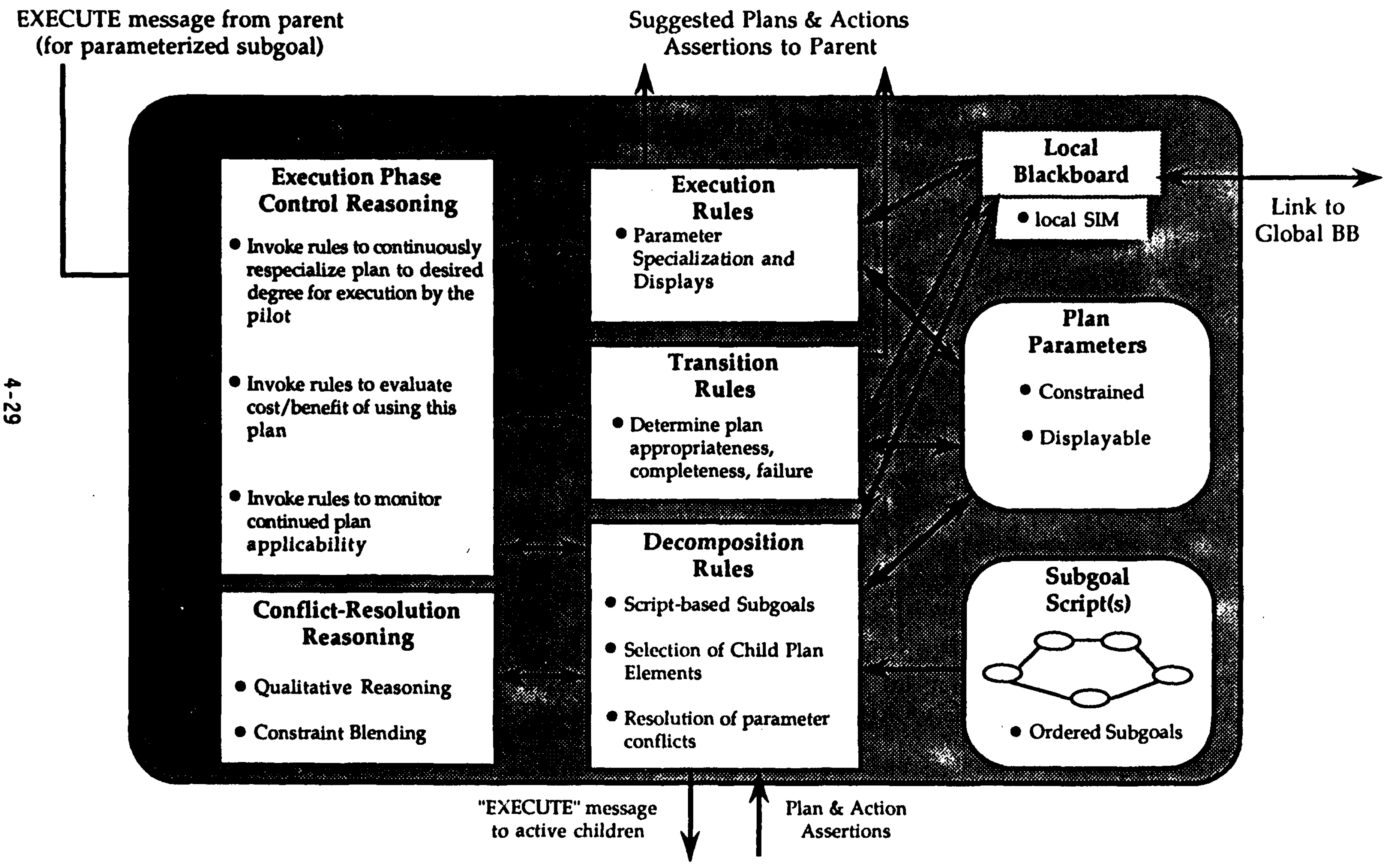

EXECUTE message from parent (for parameterized subgoal)
Suggested Plans & Actions Assertions to Parent
Execution Phase Control Reasoning
● Invoke rules to continuously respecialize plan to desired degree for execution by the pilot
● Invoke rules to evaluate cost/benefit of using this plan
● Invoke rules to monitor continued plan applicability
Conflict-Resolution Reasoning
● Qualitative Reasoning
● Constraint Blending
Execution Rules
● Parameter Specialization and Displays
Transition Rules
● Determine plan appropriateness, completeness, failure
Decomposition Rules
● Script-based Subgoals
● Selection of Child Plan Elements
● Resolution of parameter conflicts
Local Blackboard
● local SIM
Link to Global BB
Plan Parameters
● Constrained
● Displayable
Subgoal Script(s)
● Ordered Subgoals
"EXECUTE" message to active children
Plan & Action Assertions

Figure 4-9. The Execution Phase - Executing the Planning Hierarchy

for serious consideration by the planning world and needs to develop a complete set of actions. Specialization rules continue to execute, updating this element to the current situation. Transition rules evaluate the plan appropriateness, completeness or failure. Decomposition rules are used to split the problem into solvable sub-problems based on the subgoal scripts and to decide when to send execution messages to the children. Execution rules run to execute any processing that needs to happen at this level of planning, integrating standard systems into the planning process. (Typically, at the leaf nodes, there are no subgoals, no decomposition rules, and the execution ruleset resolves to basic actions that make up the planning element.)

Another way to look at the planning elements is to examine the planning element life cycle, depicted in Figure 4-10. Initially, a plan element is inactive, receiving and sending no messages and conducting no planning. When a SELECT message is received, the plan becomes a candidate. After the specialization ruleset is executed, it becomes specialized. If the selection rules determine that it is likely to achieve the relevant subgoal, it is upgraded to a feasible plan. If the parent goal decides to investigate the plan further by sending an EXECUTE message, the plan becomes active and begins to plan a complete set of actions to achieve the subgoal. The parent plan evaluates all of the active plans that satisfy a given subgoal and chooses among them, producing a selected plan. This plan is then proposed to the CO. If the CO chooses the plan, it becomes invoked. If not, it is revoked, dropping in priority. When a plan becomes active, selected, invoked, and finishes execution, it cycles back to being inactive, waiting for a new SELECT message to show that it has again become relevant.

4.2.4 Tactics Planner Phase I Functionality

The goal of the Tactics Planner software demonstration at the end of Phase I was to demonstrate the basic capabilities of the subsystem. To accomplish this goal, a narrow slice operational scenario was chosen. This scenario involved ownship making initial contact with and reacting to an enemy submarine after ownship had crossed the layer coming up for a Periscope Depth Operations (P/D Ops) maneuver.

Basic Kadet Functionality

Using Kadet enabled the Tactics Planner to immediately demonstrate many of the requirements for the full system. These included the following components:

Planning Frameworks - The system generated, maintained, optimized and executed plans.

Focus of Attention - Through the use of selection, transition, and decomposition rulesets, the Tactics Planner was able to focus attention on 'relevant' plans and options.

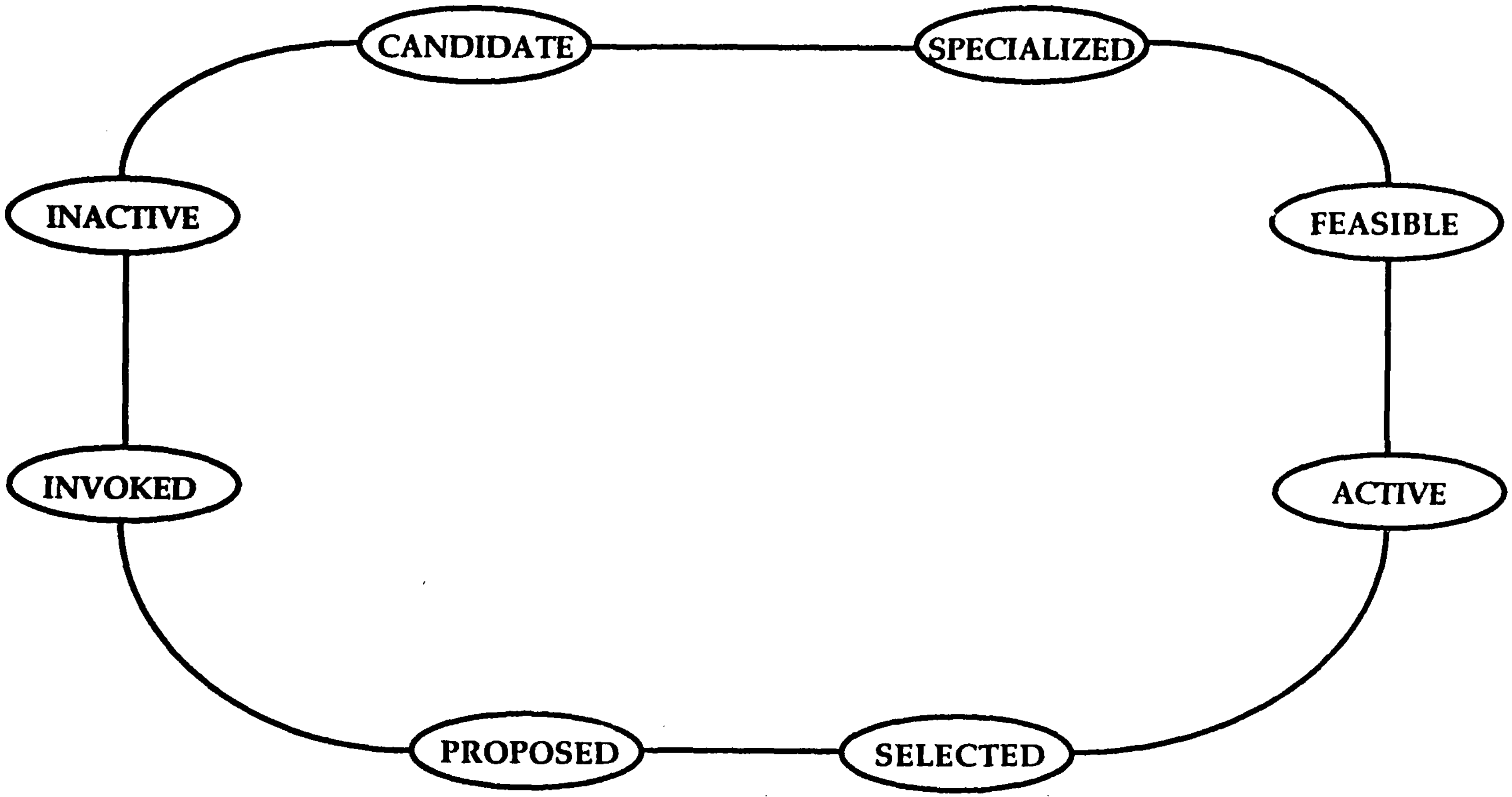


Figure 4-10. <u>The Planning Element Life Cycle</u>

Constraint Reasoning - By specializing skeletal plans according to parameters that are dynamically set on global and local blackboards, constraints to the reasoning process were used as system state variables and integrated into the normal planning process.

Constraints - Constraints become state variables which can be used like any other parameter to guide the planning process. The ability to handle domain-specific constraints, such as system status, mission plans, or information requirements, has been demonstrated. The use of Kadet and domain-specific knowledge acquisition will give the Tactics Planner the ability to handle constraint-based reasoning.

Conflict Resolution - The Kadet tool provided the Tactics Planner with two levels of conflict resolution. A more complete description of the local and global methods of handling conflicts can be found in the System Development Plan.

Parallel Planning - The Tactics Planner showed the ability to reason in parallel about multiple, possibly competing, plans and to assist the CO in selecting between them. The rulesets that focused attention to 'weed out' unimportant plans also insured that all relevant plans were considered. The plan life cycle allowed multiple, competing plans to be active, each going through the entire planning process to generate a complete set of actions. Decisions were made to select between these active plans. Each active plan was concerned with planning itself, and the decision between active plans was handled higher in the hierarchy.

Knowledge Acquisition Tools - The graphical interface to the Kadet tool was a major piece of functionality demonstrated during Phase I. The Kadet tool has developed a sophisticated, mature methodology for debugging, examining and building the knowledge base. It contained the basic TP interface with capabilities for examining the planning process, including the ability to trace the growth of the Plan and Goal planning tree, trace the rules and rulesets, examine the operational tactical situation, and monitor the message traffic. Other tools that were used in Phase I were the Plan and Goal Graph editor and the Rule/Ruleset Editing system. All of these tools are already operational and will facilitate the development of the full-scale system.

Additional Phase I Functionality

The Kadet planning tool provides much of the basic functionality. During Phase I, progress was made in other key areas as described below.

Reasoning with uncertain information - The Tactics Planner was updated to begin reasoning with uncertain information. The 'facts' of the system have been converted to structures which maintain uncertainty levels, show the source of uncertainty, and are able to trace back through the changing level of uncertainty of the fact. The rulesets in the system were augmented to show reasoning about probabilistic data by choosing between "goodness" of various plans. This decision is a domain-context-dependent arbitrator. (For example, the p(k), p(ck) and p(cd) were used in deciding between competing attack plans.)

Deferred plan selection was also demonstrated. This is the ability to defer commitment to any plan until enough information is available and the plan 'goodness' is high enough. By deferring commitment, plans to gather information and increase information reliability can be run to optimize the deferred plans.

Hypothesis "what-if" reasoning - The Kadet planner showed the capability to change the perceived values for the selection criteria for plans. The new values would possibly affect plan selection. If the new outcomes were evaluated as more accurate, the hypothesized values could replace the current values. If not, the hypothesized values could be scrapped and the original values restored.

4.3 SITUATION ASSESSMENT SUBSYSTEM

4.3.1 Overview of Situation Assessment

The Situation Assessment (SA) subsystem derives meaning from the data available about the submarine's external environment. This data includes ambiguous, incomplete, and uncertain information as it relates to the planned tactical and mission objectives of the submarine. At the highest level, SA subsystem functions can be grouped into two general categories: assessing contacts by computing and inferring additional high-level attributes about them, and monitoring the external environment for the occurrence of situations of interest.

Inputs and Outputs

Inputs and outputs to the Phase I Situation Assessment subsystem are shown in Figure 4-11. Inputs to SA come directly from the System Manager (SM) subsystem but derive from three external sources:

- The Command Interface, which transfers requests for information and monitoring activities from the Commanding Officer
- Other SOAS subsystems, including SM, which request information and monitoring activities
- The simulation environment, which provides updated information about contacts and system solutions

For the Phase I implementation, outputs from SA were sent to displays on the SA Symbolics terminal and to displays on the IRIS graphics machine. In future phases, these outputs will be sent to the Command Interface for display to key attack party members and to other SOAS subsystems for internal processing. Outputs from Situation Assessment consist of the following:

- Contact assessment, including capabilities, intent, and lethality
- Closest point of approach (CPA)

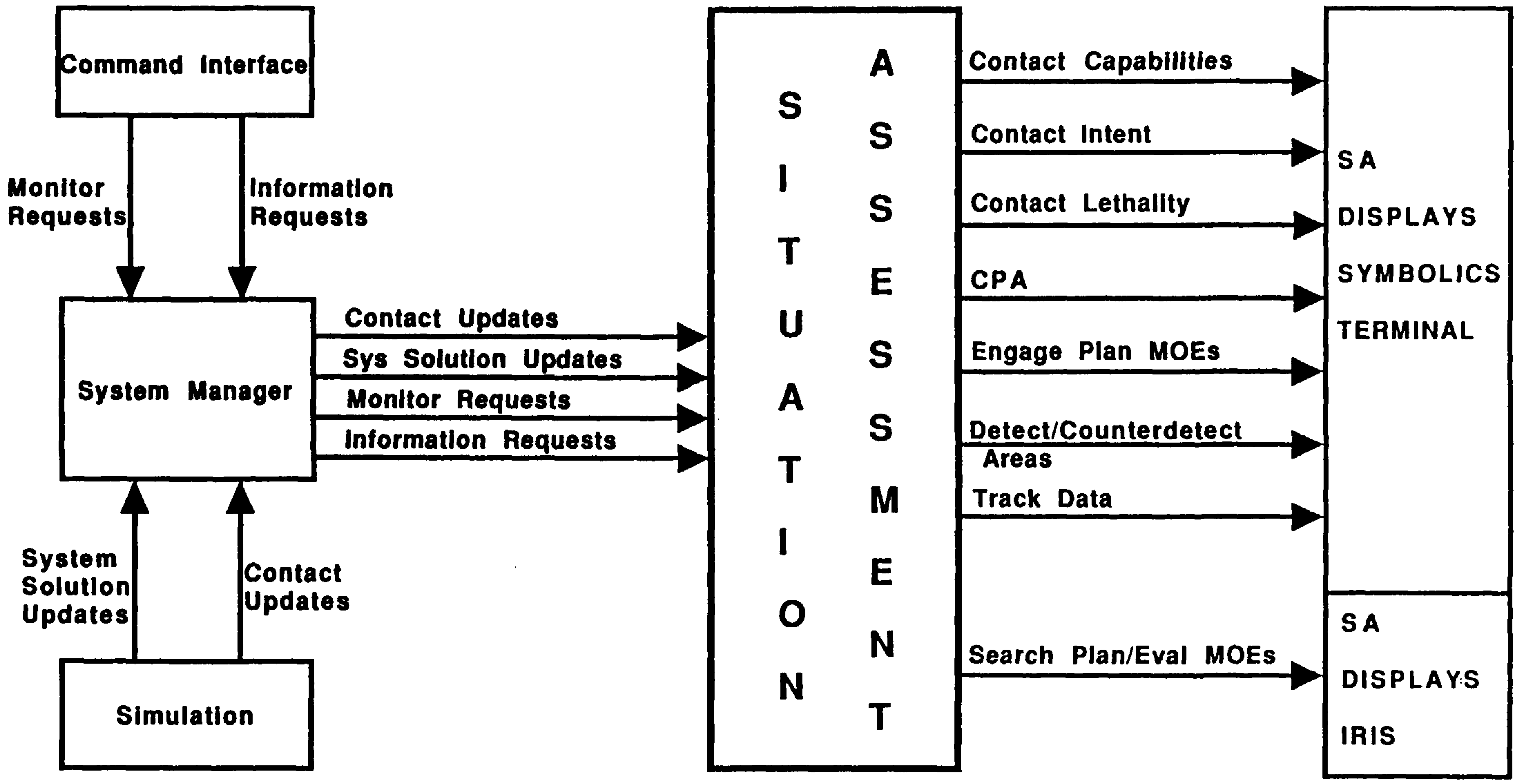


Figure 4-11. Primary SA Inputs and Outputs

- Assessment of expected engagement outcomes (measures of effectiveness)
- Areas of detection and counterdetection for use in search planning and evaluation
- Assessment of search planning and evaluation options (measures of effectiveness)
- Track data, such as bearing, bearing rate, source designator, frequency, signal-to-noise ratio, and depth elevation angle

Hardware and Tools

Situation Assessment software was implemented primarily on a Symbolics 3600 series Lisp machine using the Symbolics Genera 7.2 operating system. Source code was written in Common Lisp and used Inference's Automated Reasoning Tool (ART) which provided a rule-based system for reasoning and a blackboard-type agenda for structured control of the reasoning. In addition, portions of the SA subsystem related to expected engagement outcomes and to search planning and evaluation were hosted on a VAX 11/780 and written in FORTRAN for ease of algorithmic computations.

SA Software Design

The SA software design is shown in Figure 4-12. Current SA implementation consists of four separate processes whose primary functions are as follows:

1. **Input Data Handler** - Reads connections with the simulation, other SOAS subsystems, and the predictive algorithms for assessing engagement outcomes (which resided on a VAX 11/780)

2. **Output Data Handler** - Prepares and sends messages to other SOAS subsystems and to the predictive algorithms for assessing engagement outcomes

3. **Algorithmic Threat Attribute Computation** - Updates the tracks database with contact and mission data and computes additional algorithmic contact attributes including predictive models for probability of kill, probability of counterkill, and probability of counterdetection.

4. **Threat Monitor and Information Stream Management** - Monitors the tracks database for specific conditions of interest and generates messages to notify the SOAS system of these conditions.

Communications between the I/O processes and the other two processes is accomplished using message queues (Input Agenda, Output Agenda, and Monitor Agenda). The Algorithmic Threat Attribute Computation and Threat Monitor and Information Stream Management processes communicate through common access to the SA Database.

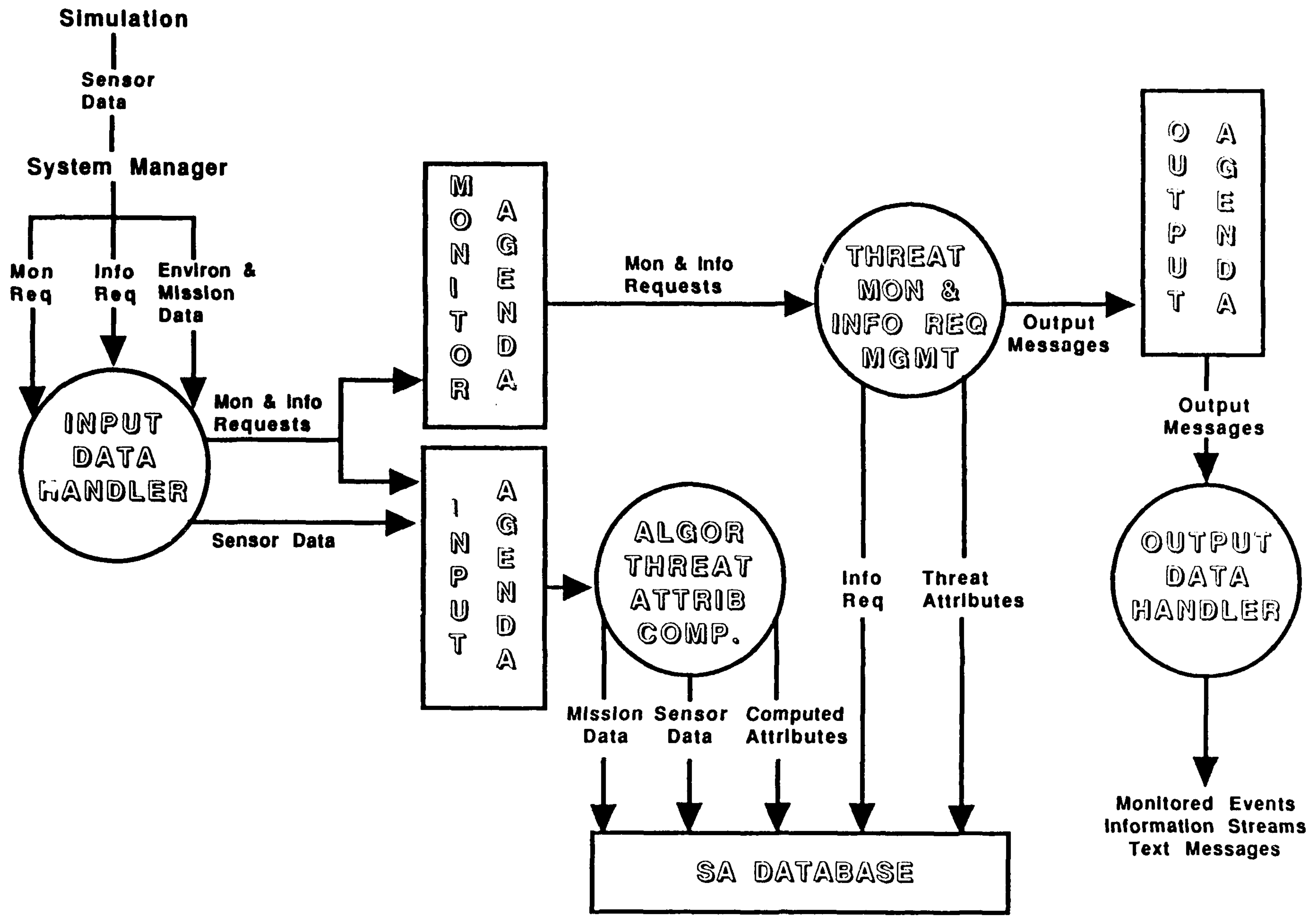


Figure 4-12. SA Software Design

SA functionality has been divided into four modules as seen in Figure 4-13: Database Maintenance, Contact Assessment Control, Contact Definition, and Contact Monitoring. These SA subsystem modules are described in the following four sections. The description of Situation Assessment concludes with a summary of the SA Database.

4.3.1 Database Maintenance Module

The Database Maintenance module accepts track file data from the simulation (with assumed sensor fusion) and mission data from other SOAS subsystems, and stores it in the appropriate database. It also computes additional geometric and algorithmic attributes, as specified by the Contact Assessment Control module, which are in turn used by the Contact Definition module.

Upon initialization, a process is created to generate and read the connection with the SM subsystem, through which all message traffic flows including simulation data and MOEs for engagement planning. Messages are received from the SM and placed on an input queue. A separate process then parses the messages and stores the data in the appropriate database. Additional algorithmic contact attributes are computed as specified by the monitoring and information requirements of Contact Assessment Control.

Upon request by Tactics Planner, Database Maintenance module assesses the expected engagement outcomes for a given scenario by computing measures of effectiveness (MOEs). MOEs for engagement planning include the following:

- Probability of detection/counterdetection
- Probability of track/countertrack
- Probability of kill/counterkill
- The exchange ratio for each attack

Lisp functions are used to read messages from the external connections and some algorithmic attribute computations, along with FORTRAN. The input queue and databases are implemented using Lisp Flavors.

Knowledge required by this module to provide the algorithms for computing additional contact attributes can be classified into two categories: simple physics and geometry knowledge, and SA domain knowledge. Physics and geometry knowledge has been acquired primarily from reference books. SA domain knowledge has been obtained from the domain experts at Kapos, Presearch, and Search Technology. Domain knowledge acquisition has taken place through interviews with the experts, reports written by the experts, other domain literature, and algorithms generated by the experts.

Currently computed algorithmic contact attributes include, in part: time to intercept, weapon envelopes, launch acceptability regions, sensor envelopes, probability of kill, probability of counterkill, probability of counterdetection, and CPA.

The predictive calculations for probability of kill/counterkill are computed as functions of weapon and target characteristics, range at time of torpedo launch, angle off bow, the accuracy of the firing solution, alert-

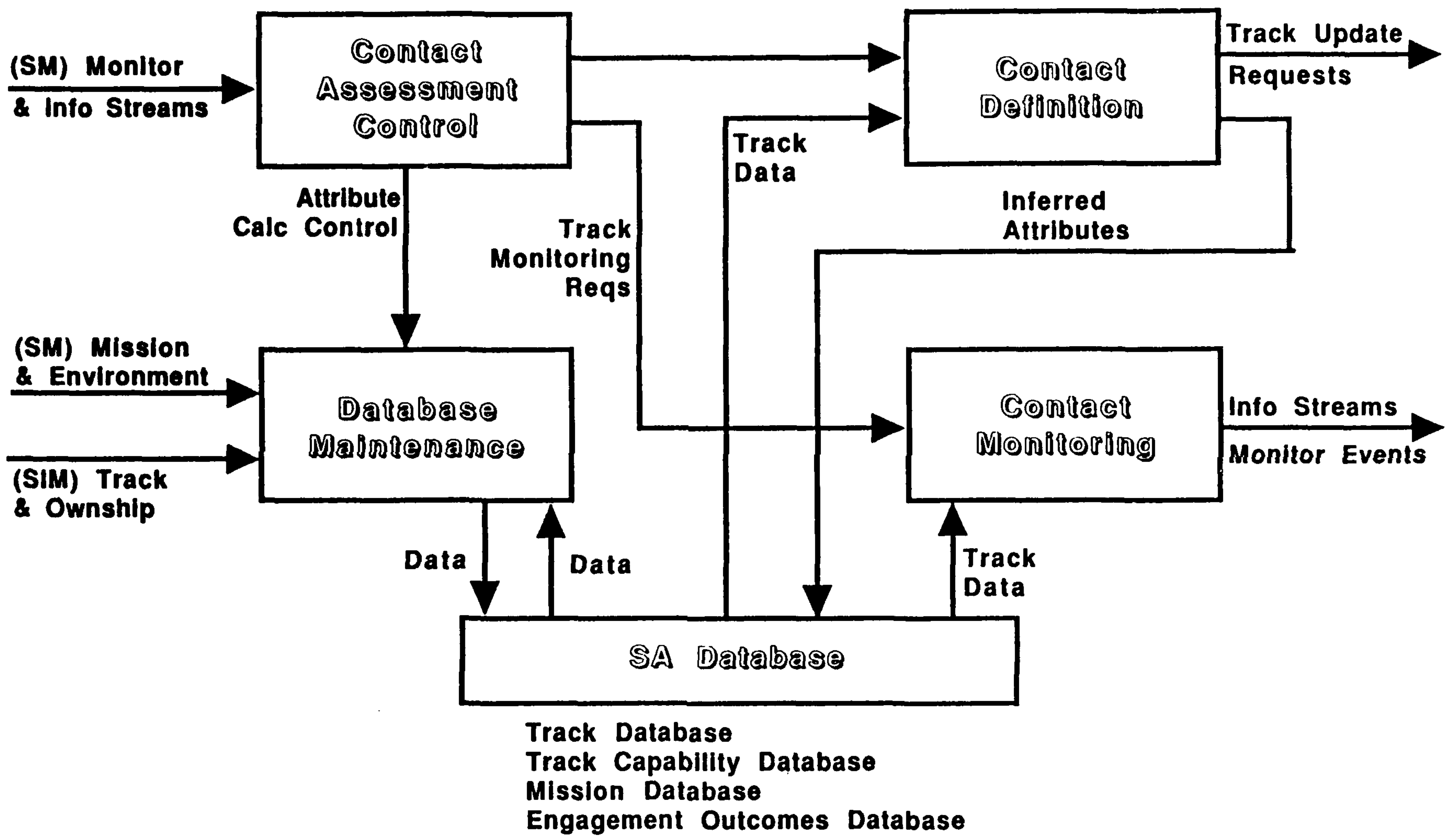


Figure 4-13. SA Functional Modules

ing event (launch or enable), and evasion tactic (including countermeasures). The predictive calculations for probability of counterdetection are computed as functions of the SVP, FDM parameters including source level as a function of AOB and speed, and short- and long-term acoustic variation. The simulation facility provides the data for these calculations. The idea of these functions is not to merge all available data and related uncertainty into one number, such as exchange ratio, but rather to segment the tactically important components such as counterdetection, TMA, and weapon/counterweapon capability in the face of countermeasures and, thus, to give the commander the option of viewing a variety of aspects of his position or just the one number, if he so chooses. Additional details about the predictive algorithms used by SA are provided in Section 4.3.6.

Figure 4-14 details a Lisp Flavor that represents all the information currently contained in a track for the SOAS system for Phase I.

4.3.2 Contact Assessment Control Module

The Contact Assessment Control module provides a mechanism by which the other SOAS subsystems can request SA track information or establish monitors to detect situations of interest. In addition, this module controls the assessment and computation of both algorithmic and heuristic contact attributes based on the current priority and information requirements of specific tracks.

Two types of requests are handled by Contact Assessment Control:

1. **Monitor requests** - Generated by Tactics Planner and sent via System Manager to monitor for specific situations of interest (events) based on the current activities of the commander and SOAS system

2. **Information requests** - Generated by Command Interface or Tactics Planner and sent via System Manager to supply the requesting subsystem or the Display Generator with contact attributes (information streams) based on the information requirements of the currently active plans

For the Phase I implementation, specific monitor and information requests were detailed in the SOAS Interface Control Document and are discussed further in the Contact Monitoring section.

Lisp functions were used to receive [illegible] parse the request messages. Monitor and information requirements wer[illegible]mented using ART rules and its blackboard structure.

Knowledge which defines monitors and information streams was acquired through analyzing the purpose and information requirements of the plans used in the SOAS system. Details regarding the content of the monitors and information stream were developed through discussions with the subsystem leads whose subsystems were responsible for generating the plans which use the information. Assessment control knowledge within this module also exists in the form of knowing to what level the assessment of a contact is

```
;;; Current attributes for all tracks in Phase I.
;;; Each instance of this track flavor will have these attribute slots.
;;; Values for these attributes are either calculated, inferred, prebriefed,
;;; or Unknown and are updated dynamically.
;;; Some values may be inappropriate for certain tracks and therefore are left
;;; as an empty slot.

(scl:defflavor TRACK
        ((PRE-BRIEFED nil)
         (UPDATE-RATE '(120))
         (TRACK-NUMBER (scl:make-instance 'slot))       ;;; SA object number (integer)
         (SENSOR-ID (scl:make-instance 'slot))
         (TIME-FIRST-SEEN (scl:make-instance 'slot))    ;;; mission time, seconds (integer)
         (SENSOR-AGE (scl:make-instance 'slot))
         (SYSTEM-AGE (scl:make-instance 'slot))
         (SONAR-MODE (scl:make-instance 'slot :value 'unknown)) ;;; acoustic mode for now (passive,
                                                        ;;; active,acquired,hit,exhausted)
         (POSITION-LONG (scl:make-instance 'slot :value 0))     ;;; degrees, longitude (real) <xdisp>
                                                        ;;; calc-ed from range in yards
         (POSITION-LAT (scl:make-instance 'slot :value 0))      ;;; degrees, latitude (real) <ydisp>
                                                        ;;; in parse-message
         (POSITION-DEPTH (scl:make-instance 'slot :value 0))    ;;; feet, below MSL (real) <zdisp>
         (COMMANDED-DEPTH (scl:make-instance 'slot :value 0))   ;;; icd 4-14
         (DEPTH-RATE (scl:make-instance 'slot :value 0))        ;;; icd 4-14 (dive-rate)
         (SPEED (scl:make-instance 'slot :value 0))             ;;; knots (real)
         (COMMANDED-SPEED (scl:make-instance 'slot :value 0))   ;;; icd 4-14-89
         (ACCELERATION (scl:make-instance 'slot :value 0))      ;;; icd 4-14
         (THRUST (scl:make-instance 'slot :value 0))            ;;; icd 4-14
         (COMMANDED-THRUST (scl:make-instance 'slot :value 0))  ;;; icd 4-14
         (DIVE-ANGLE (scl:make-instance 'slot :value 0))        ;;; icd 4-14
         (COURSE (scl:make-instance 'slot :value 0))            ;;; (HEADING) degrees (real)
         (COURSE-RATE (scl:make-instance 'slot :value 0))       ;;; deg/min (real) (heading-rate)
         (COMMANDED-COURSE (scl:make-instance 'slot :value 0))  ;;; icd 4-14
         (RUDDER (scl:make-instance 'slot :value 0))            ;;; icd 4-14
         (COMMANDED-RUDDER (scl:make-instance 'slot :value 0))  ;;; icd 4-14
         (SENSOR-RANGE (scl:make-instance 'slot))               ;;; nm, relative range from ownship
         (SENSOR-RANGE-RATE (scl:make-instance 'slot))          ;;; nm/sec2 (real)
         (SENSOR-RANGE-SD (scl:make-instance 'slot))
         (SYSTEM-RANGE (scl:make-instance 'slot))
         (SYSTEM-RANGE-RATE (scl:make-instance 'slot))          ;;; nm/sec2 (real)
         (SYSTEM-RANGE-SD (scl:make-instance 'slot))
         (REL-BEARING (scl:make-instance 'slot))                ;;; degrees (real)
         (SENSOR-BEARING (scl:make-instance 'slot))             ;;; degrees (real)
         (SYSTEM-BEARING (scl:make-instance 'slot))
         (BEARING-INIT (scl:make-instance 'slot))
         (SYSTEM-BEARING-RATE (scl:make-instance 'slot))        ;;; deg/min (real)
         (SENSOR-BEARING-RATE (scl:make-instance 'slot))        ;;; deg/min (real)
         (SENSOR-BEARING-SD (scl:make-instance 'slot))
         (FFN (scl:make-instance 'slot :value 'unknown))        ;;; see ICD Appendix A (integer)
         (FFN-PROBABILITY (scl:make-instance 'slot))            ;;; 0 to 100 (real)
         (CLASS (scl:make-instance 'slot :value 'unknown))      ;;; see ICD Appendix A (integer)
         (CLASS-PROBABILITY (scl:make-instance 'slot))          ;;; 0 to 100 (real)
         (TYPE (scl:make-instance 'slot :value 'unknown))       ;;; see ICD Appendix A (integer)
         (TYPE-PROBABILITY (scl:make-instance 'slot))           ;;; 0 to 100 (real)
         (TORPEDO-LAUNCH-DETECTION-FLAG (scl:make-instance 'slot))      ;;; 0-no launch, 1-missile
                                                                ;;; launched 2-torp engine det
                                                                ;;; 3-torp enable detect
         (TORPEDO-LAUNCH-TERMINAL-FLAG (scl:make-instance 'slot))       ;;; 0 - no explosion, 1 - missile
                                                                ;;; exploded (integer)
         (DAMAGE-ASSESSMENT (scl:make-instance 'slot))          ;;; 0 - N/A, 1 - destroyed, 2 - damaged,
                                                        ;;; 3 - missed (integer)
```

Figure 4-14. SA Track Information (Sheet 1 of 2)

```
        (object-tracking (scl:make-instance 'slot))
        (range-at-launch (scl:make-instance 'slot))
        (critical-target (scl:make-instance 'slot))
        (target-of-concern (scl:make-instance 'slot))
        (critical-object (scl:make-instance 'slot))
        (object-of-concern (scl:make-instance 'slot))
        (critical-surface (scl:make-instance 'slot))
        (surface-of-concern (scl:make-instance 'slot))
        (critical-submerged (scl:make-instance 'slot))
        (submerged-of-concern (scl:make-instance 'slot))
        (critical-airborne (scl:make-instance 'slot))          ;;; ASW patrol or helo
        (airborne-of-concern (scl:make-instance 'slot))
        (critical-torpedo (scl:make-instance 'slot))
        (torpedo-of-concern (scl:make-instance 'slot))
        (shortest-time-to-lar-against-ownship (scl:make-instance 'slot))
        (optimum-shortest-time-to-lar-against-ownship (scl:make-instance 'slot))
        (shortest-time-to-intercept (scl:make-instance 'slot))
        (time-to-intercept (scl:make-instance 'slot))         ;;; CPA
        (time-to-lar-against-ownship (scl:make-instance 'slot))
        (time-to-lar-against-object (scl:make-instance 'slot))
        (shortest-time-to-lar-against-object (scl:make-instance 'slot))
        (optimum-shortest-time-to-lar-against-object (scl:make-instance 'slot))
        (threat-value (scl:make-instance 'slot))
        (threat-value-mission (scl:make-instance 'slot))
        (target-value (scl:make-instance 'slot))
        (aspect-angle (scl:make-instance 'slot))       ;;; degrees, ownship relative to object (real)
        (zig-detect (scl:make-instance 'slot))
        (speed-change (scl:make-instance 'slot))
        (course-change (scl:make-instance 'slot))
        (aspect-angle-change (scl:make-instance 'slot))
        (SNR-value (scl:make-instance 'slot))
        (pcd-est (scl:make-instance 'slot))                    ;;; his ability to detect us
        (pch-est (scl:make-instance 'slot))                    ;;; his ability to hit us
        (ph-est (scl:make-instance 'slot))                     ;;; his ability to be hit by us

        (active-int (scl:make-instance 'slot))
        (transient-det (scl:make-instance 'slot))
        (OS-transient-det (scl:make-instance 'slot))
        ;;(intent (scl:make-instance 'slot))
        ;;(intent-mission (scl:make-instance 'slot))
        (capability (scl:make-instance 'slot))
        ;;(WISOI (scl:make-instance 'slot))
        (torpedo-target (scl:make-instance 'slot))
        (deamon nil)
        )
       (DISPLAY-ICON)
   :readable-instance-variables
   :writable-instance-variables
   :initable-instance-variables)
```

Figure 4-14. **SA Track Information** (Sheet 2 of 2)

required. This knowledge is responsible for determining which contact attributes should be calculated and when the calculations should be performed.

4.3.3 Contact Definition Module

The Contact Definition module infers high-level attributes about contacts and will handle the uncertainty associated with them. Based on the information and assessment requirements given by Contact Assessment Control, Contact Definition heuristically determines qualitative features of a contact and handles Phase I level corrupt information.

The functions in this module which infer contact attributes use rules to perform their assessments. These rules utilize contact information provided by the Database Maintenance module in the Tracks Database and information in the Track Capability Database which contains specific knowledge regarding the capabilities of threats (subs, ships, ASW patrols, torpedoes). Other information used would be any pre-briefed information, communicated information, commander's intuition, rules of engagement, and environment conditions (SVP).

Because of the qualitative nature of these assessments, it is essential to have a strategy for handling uncertainty. During Phase I, theoretically based probabilistic functions were used to handle the uncertainty in the data for determining predicted probability of kill, probability of counterkill, probability of counterdetection, and other target solution parameters such as range, which are derived by course and speed emulations in the simulation's TMA module. More advanced methods will be added to the stochastic approach in later phases to help determine attributes such as probable intent of threat, stronger threat identification.

4.3.4 Contact Monitoring Module

The Contact Monitoring module monitors the SA database for situations of interest based on the planning activities of the commander and thus the SOAS system. It also distributes contact information based on the information requirements of the other SOAS subsystems.

The SA has a set of predefined monitors which are invoked and revoked according to the currently active plans within the SOAS system. These monitors are used to detect the occurrence of specific events in the external environment which can signify the success or failure of active plans or identify a situation that might require new planning.

Contact information distribution is performed using information streams. Information streams are essentially data streams containing a predetermined list of contact attributes which are turned on based on the needs of the SOAS planning subsystems. Depending on the type of data being sent, information stream messages may be updated synchronously with a variable update rate, only once, or asynchronously when attributes change. The purpose of information streams can be summarized as supplying the right information, to the right subsystem, at the right time, and at the desired update rate.

The majority of the monitoring functions in this module are implemented in ART rules. Contact Assessment Control attaches values to attribute slots in the SA database. ART rules fire whenever a change in a slot value triggers a match for some fact in the rules. These rules then call Lisp functions that check for conditions of the monitor by accessing relevant contact attributes. The broadest and most inexpensive delimiting conditions are initially checked, leaving the conditions with more expensive computations to be done only after the initial conditions are satisfied. When all conditions of a monitor have been met, a message is generated notifying the appropriate SOAS subsystem that the monitored event occurred.

Information streams are implemented in much the same way as the SA monitors. As active values change, Lisp functions are called which construct the appropriate information stream message, accessing previously computed attributes and calculating any other contact attributes required for the message.

The knowledge used in the Contact Monitoring module has been acquired through analysis of the SOAS Plan-Goal Graph and discussions with other subsystem leads. These discussions are used to define the information requirements for plans, as well as describe how to perform appropriate contact monitoring for active plans.

4.3.5 The SA Database

The SA database for Phase I can be divided into four sections:

- **Track database** - The track database is an object-oriented representation of information known about each contact in the environment, implemented using Lisp Flavors. These contacts are represented as tracks with some 40 attributes for each track, including a separate track for ownship. These tracks, which have already been correlated from various sensor contacts by a sensor fusion function, contain any pre-briefed, sensed, computed, communicated, and inferred information about all contacts in the environment, friendly or enemy. (This sensor fusion function is simplified for Phase I, but will be used in later phases to correlate many sensor platforms, for which SA will enhance the output by reasoning more deeply about the surrounding environment and all ambiguous and incomplete data).

- **Track capability database** - The tracks will also contain information as to the capability of the threat. This data will be obtained from a capability database, which for Phase I contained weapons carried, weapon ranges, and sensor ranges based on published information in Jane's Fighting Ships.

- **Mission database** - This portion of the SA database contains information about the current mission, ownship status, and briefed information, such as airbase locations, SOSUS arrays, and communications zones.

- Engagement outcomes database - Unlike the previous three databases which resided on the SA Symbolics machine, the databases and models related to assessing engagement outcomes were hosted on the Vax 11/780 for ease of computation. The following section describes the models which were used as well as the algorithms for predicting the outcomes of submarine engagements.

4.3.6 Assessment of Expected Engagement Outcomes

The paragraphs which follow describe the methodology for the predictive algorithms and databases used by the Database Maintenance module to assess the outcome of submarine engagements and the history of acceptance this methodology has enjoyed in the Naval community. The specific equations used to model submarine encounters are discussed first. Next is a brief description of the numerous models necessary to perform these calculations. The models include weapons models, propagation loss models, search planning and evaluation models, time-motion analysis (TMA) models, and countermeasure models. During Phase I, these models were used off line to provide databases for Situation Assessment In subsequent phases, it is anticipated that these models will be used both on and off line to provide information.

Algorithms for Computing MOEs for Expected Engagement Outcomes

The algorithms used to determine probability of detection/counterdetection, probability of track/countertrack, probability of kill/counterkill, and the exchange ratio for each attack are the same as those developed for the submarine engagement model SUBSUB.

Originally, SUBSUB was developed specifically as a high-speed, cost-effective model for use in support of the ASW POM Appraisal. It was developed in 1981 at the request of PM-4, presently SPAWAR, as an augmentation to SIM II against which it was successfully launched in the POM-84 Appraisal.

Since its introduction in the POM-84 Appraisal, SUBSUB has been utilized in several efforts and upgraded significantly to represent new tactics and systems. It was used in all the POM appraisal efforts since POM-84. In 1986, an interactive engagement was developed for the war gaming efforts at the Naval War College. To date, this model is still being upgraded and maintained for this purpose.

In 1987, the model was used in by Team A/ASW Appraisal. Following its use in Team A, SUBSUB was again reviewed. It was reviewed by CNA at the request of OP-951 as part of a broad effort to upgrade the fidelity f the ASW Appraisal models. This review indicated several areas where improvements were required to enhance the model's fidelity and to provide a more formal mathematical basis for the overall model structure.

Presearch Inc. and Metron Inc. were jointly tasked to develope an approach to implement the recommended improvements while maintaining the desirable features of the existing model, i.e., rapid turn-around time and the flexibility to represent a broad spectrum of weapon, sensor, platform, environmental and tactical inputs. An acceptable approach was developed by Presearch and Metron, approved by CNA, and implemented by Presearch.

The algorithms developed in this last review provided the basis for the quantitative assessment of systems and platforms for SOAS.

The general approach for modeling SSN one-on-one engagements in SOAS is to represent a submarine engagement as a series of independent probabilistic events and to combine them using Markov Chain techniques. The probability of occurrence for each engagement event is mathematically represented by a definite integral that includes terms for the signal excess and the kinematic conditions. For example, the ability of a platform to detect prior to being counterdetected is a function of the detection sensor performances and search tactics of the two opposing platforms. Once the individual events are determined, they are combined using Markov Chain processes to yield engagement results. Events are combined according to a predefined event sequence. This multiplication process is based on the assumption that each event is independent of the previous event but there is range dependence between events.

For example, consider the sample event probability tree given in Figure 4-15. In this tree, Blue has detected an enemy platform. When Blue proceeds to launch an attack on Orange, six events are possible:

1. $F\overline{CD}$: Blue launches a weapon; Orange has not counterdetected

2. $CD\overline{F}$: Blue is counterdetected prior to weapons launch

3. $F\overline{CT}$: Blue is counterdetected prior to weapons launch but launches a weapon prior to countertrack

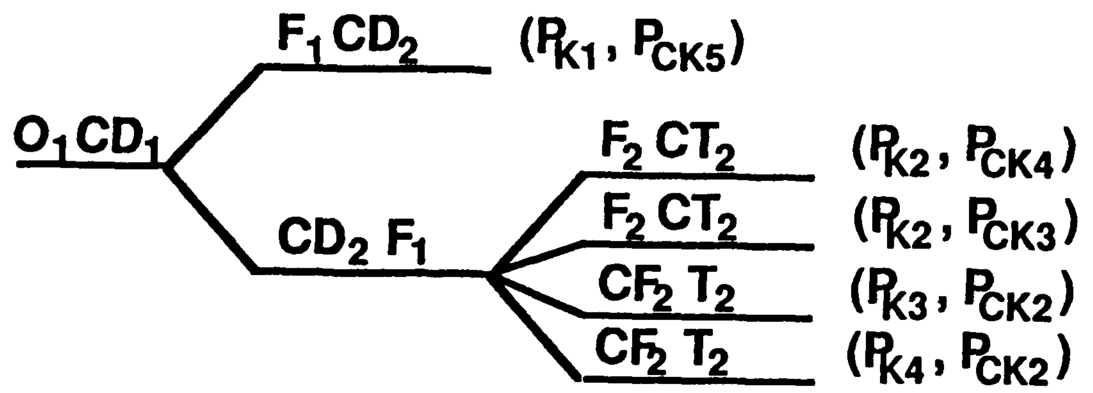
4. FCT: Blue is counterdetected and then countertracked but launches a weapon prior to counterfire

5. $CF\overline{T}$: Orange is detected prior to weapons launch but launches a weapon prior to Blue track

6. CFT: Orange is detected and then tracked but launches a weapon prior to Blue fire

From this tree, consider that a U.S. platform has detected an enemy submarine. The probability that the U.S. platform is able to launch a weapon without sacrificing covert operations, i.e. not be counterdetected, and kills the enemy platform is determined using equation presented in Figure 4-16. In this equation, PCD is the cumulative probability of counterdetection at range R and zero is the probability density for Blue's track. These probabilities are determined using sensor's Figure of Merits (FOM) and propagation loss data. The term PK is the probability of a Blue kill as a function of firing range.

Using the approach given in the above equation, it is possible to model several possible current and future one-on-one engagements scenarios. First, it allows the use of active as well as passive in all phases of an engagement. This is possible since no platform is constrained to using the same sensor for all phases of an engagement. Second, it allows the full parameterization of weapon systems. Third and final, the algorithms have



INITIAL CONDITION:
Blue Detection, Orange Unalerted

Weapon Exchange Conditions:

P_{K1} = Secure	: Uses Secure TMA with ROL_1 Against an Unalerted Target
P_{K2} = Nonsecure	: Uses Nonsecure TMA with ROL_2 Against an Alerted Target
P_{K3} = Responsive	: Uses Nonsecure TMA Each wxxxx Lxxxx rxxx Against Pxxxx Evasion
P_{K4} = Reactive	: Uses LOB to Target xxxxxxxx
P_{K5} = Snapshot	: Uses LOB to Weapon xxxxxxxx

ROL_1	= Preferred Secure Launch Point
ROL_2	= Preferred Nonsecure Launch Point
LOB	= Line of Bearing

Figure 4-15. Sample Engagement Event Tree

$$P_E \equiv P_r \left\{ ENG \rightarrow O_1 \overline{CO}_1 \longrightarrow CO_2 \bar{F}_1 \rightarrow F_2 \overline{CT}_2 \right\} \quad E_1 = (B, O) \quad E_2 (O, B)$$

$$E_3 = (B, OT)$$

$$\textbf{P (k) secure} = \int_0^{r_0} \left\{ \int_0^{r_1} \left[\int_0^{r_2} \phi_{BT_2}(r_3)\, F^3 \left(f_2(r_3) | r_2\right) PK2\left(f_2(r_3)\right) \bar{P}_{OT_2}\left(f_2(r_3)\right) dr_3 \right.\right.$$

$$\left. + P_{BT_2}(r_2)\, F^3 \left(f_2(r_2) | r_2\right) PK2\left(f_2(r_2)\right) \bar{P}_{OT_2}\left(f_2(r_2)\right) \right] \phi_{OO_2}(r_2)\, F^2 (r_2 | r_1)\, \bar{P}_{BF_1}(r_2)\, dr_2$$

$$\left. + P_{OO_2}(r_1)\, \bar{P}_{BF_1}(r_1) \right\} \phi_{BO}(r_1)\, F^1 \left(r_1 | r_0\right) \bar{P}_{OO_1}(r_1)\, dr_1$$

Where:	B = Blue	O = Orange
	D = Detect	T = Track
	1 = Secure Tactics	2 = Nonsecure Tactics

0_{xx} are probability density functions

P_{xx} are cumu' :ive probability functions

$F^n(*/r_{n-1})$ = cumulative CPA distribution for the tactics associated with the nth branch

$f_x(r_m)$ = firing range associated with tracking range r_m

$PK2(r_m)$ = weapon effectiveness a+ range r_m (condition 2 = nonsecure attack)

Figure 4-16. **Computation of One Engagement Event Branch**

the potential for future enhancements. These upgrades are necessary to model future system, platform, and tactical improvements.

Weapons Models

The weapons effectiveness data used for predicting outcome of submarine engagements is determined by one of two models, NTORP or MSTORC. Both of these models developed by Presearch Inc., are used extensively to determine weapon performance. These models generate the results found in the U.S. and Soviet Weapons Effectiveness Baseline; these two documents provide a single source of weapons effectiveness estimates intended for use in conducting studies and analyses by the Navy.

The model NTORP determines the effectiveness of torpedoes in the runout search mode. The probability of a runout torpedo hit is determined by first defining the firing ship minimum and maximum lead angles that will result in the interception of the actual target track by the actual torpedo acquisition angle. Figure 4-17 shows the minimum and maximum torpedo lead angles that will result in target acquisition opportunity. The probability that the torpedo actually acquires the target is then determined by the distribution of lead angles, which results from combining all significant platform, sensor, fire control, and launch errors with torpedo gyro errors.

The model, MSTORC, used to determine circle search torpedo effectiveness is applicable to both air-dropped torpedoes and missile delivered torpedoes (i.e., ASROC, VLA, SEA LANCE). The model is comprised of two distinct sections: the development of an acquisition and closure matrix, and the development of a splash point distribution matrix. The product of these two matrices provides the probability of hit given attack for each threat/ scenario combination.

The generation of the acquisition and closure matrix utilizes the characteristics of the weapon and the threat platform. The threat is assumed to be in the center of an NxN size matrix at the time the weapon hits the water, modeled as the origin in a Cartesian coordinate system. Subsequent movement by the threat within the coordinate systems will be dictated by the speed, acceleration rate, and evasion tactics for the submarine. For each point in the matrix, the probability that the weapon can acquire the threat is determined (averaged over a uniform distribution for initial weapon orientation). Acquisition is based on a three dimensional environment where the acoustic search beams for the torpedo provide both vertical and azimuthal coverage.

The acquisition range for a particular scenario is impacted by the submarines target strength, target doppler, target aspect, and the torpedo sonar performance characteristics. Upon acquisition, the ability of the weapon to close the target is driven by the range to the target, the relative speeds of the target and torpedo, and the endurance of the weapon.

The generation of the splash point distribution matrix is driven from the ability of the attacking platform to localize the threat and to deliver the

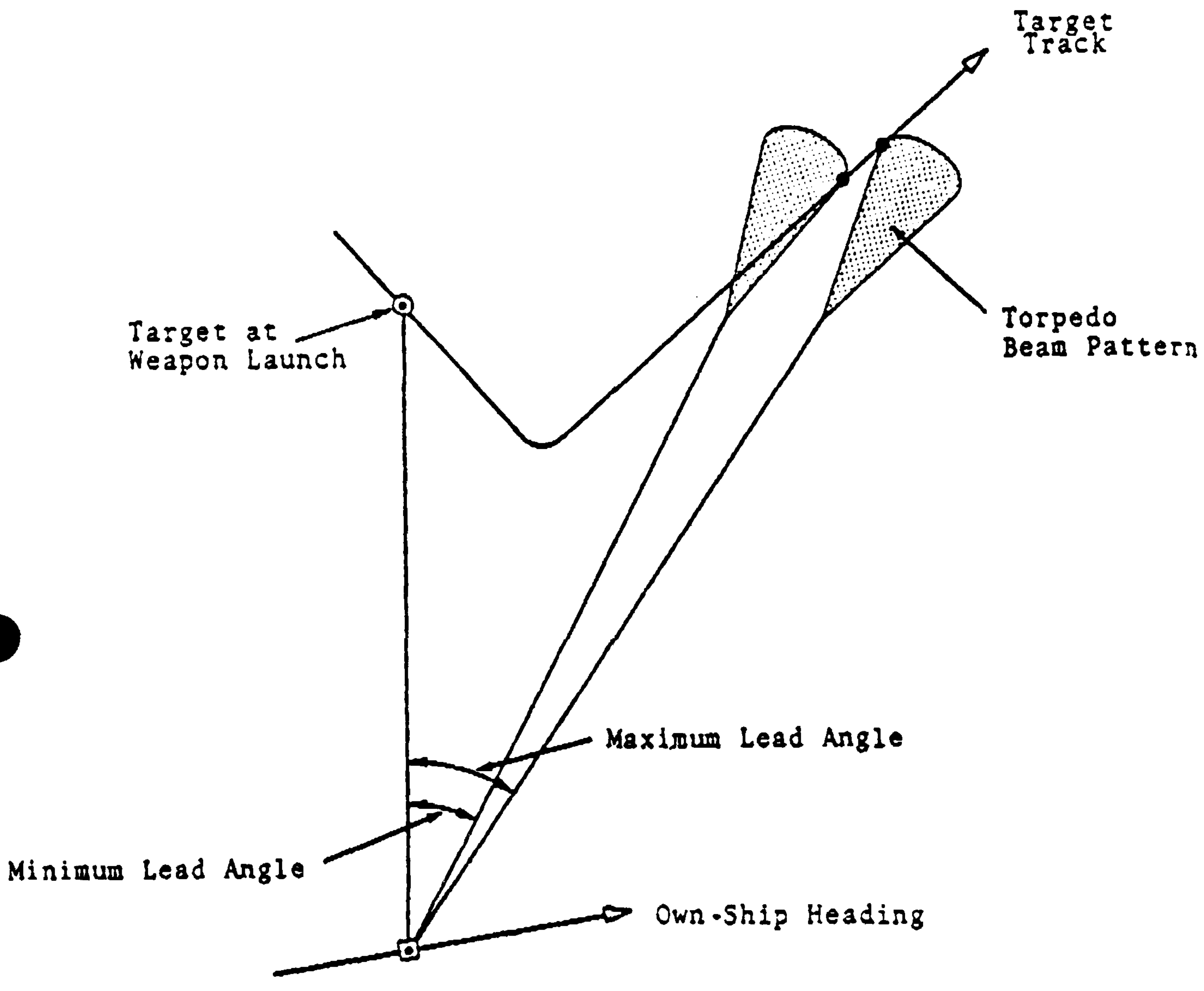


Figure 4-17. Geometry to Determine Lead Angle Limits

weapon to the right point. The model is designed to compute two sigma values and two bias values for down range and cross range errors. These sigmas and biases are determined for each specific scenario and firing range, and are impacted by the type of sonar system and technique being used to perform the target motion analysis (TMA). The sigma values are used to determine the probability of a weapon splashing at each point on the matrix with the center of the matrix corresponding to the origin of a Cartesian coordinate system. The biases will shift the location of the origin of the splash point distribution matrix relative to the origin of the acquisition and closure matrix. The subsequent product of the two matrices provides the probability of hit given attack for the scenario.

Representative answers for weapons effectiveness questions were collected for this Phase I effort and stored in a database for referencing by the Situation Assessment module. For this initial database, the parameters considered (and thus the indices of the database) were the targets AOB and the range to the target. There are many degrees of freedom left to exploit remaining in the model for a more complete database.

The models consider the effects the following major input parameters that might be used as parameters for predictive databases:

- Range
- Initial AOB
- Downrange and crossrange placement error (Solution quality) Target alertment and maneuvers
- Target initial speed
- Weapon/target depths
- Vertical and horizontal beam pattern

For those parameters for which there is partial or no information to consider, assumptions may be made or averages computed.

Environmental Model

The environmental data used for predicting the outcome of submarine engagements considers a wide variety of parameters such as the sound transmission loss data and ambient noise as a function of geographical locations. The propagation loss data was generated using the Generic Sonar Model (GSM) and ambient noise was obtained from the Acoustic Baseline.

The Generic Sonar model was developed as a joint effort between NUSC and NOSC at the request of POSSM, the Naval Sea Systems Command 06H1 Panel On Standard Sonar Models. The model was developed to aide in the design of sonar systems. Along with providing the necessary transmission loss data for the equations used to predict engagement outcomes, the model was designed to determine passive and active signal excess as a function of range frequency and source/receiver depths, to perform LOFAR diagrams, Autocorrelation functions and cross-correlation functions.

The Acoustic Baseline (ABL), originally written in 1973, provides a common standard reference for worldwide environmental data provided by the Naval Oceanographic Research and Development Activity (NORDA). The document is continually upgraded and revised in an attempt to represent the complexity of oceanographic and meteorological conditions that affect underwater acoustics. It contains sound velocity profiles and ambient noise for all geographical areas.

4.3.7 Search Planning and Evaluation

Prior to engagement, Situation Assessment can optimize the search planning and underway search evaluation phases of ASW operation by determining the detection/counterdetection areas and computing measures of effectiveness. Candidate MOEs for search planning and evaluation include the following:

- Sweep rate
- Probability of detecting first
- Probability of secure detection
- Average and minimum probability of detect
- Daily engagement work
- Exchange ratio
- Probability of kill

The probability of detecting first provides for considering the two-sided nature of the problem and requires algorithms for predicting counterdetection capabilities. The probability of kill along with the exchange ratio allows the CO to evaluate the risk associated with achieving the mission objective of killing the target, and to tailor that risk to the operational scenario. (In the event that the SSN is performing area clearing search for a CVBG that is to arrive soon, the CO should be willing to risk more.) Sweep rate will be calculated using the integrated effect of all of the search capability, not just a cookie-cutter representation times ownship speed.

The methodology for optimizing search planning and underway search evaluation was developed by Presearch, Inc. Although this functionality was not integrated into the SA subsystem for Phase I, data similar to the results of running these algorithms was presented in the notional displays during the SOAS Phase I demonstrations.

The methodology was developed and a prototype implemented as a part of the BSY-II proposal effort. The modeling approach utilizes this proven, straightforward methodology that readily incorporates critical realistic features of ASW operations including vertical/horizontal asymmetries; target motion; short and long term signal excess; continuous and transient target/ ownship noise sources; constant speed or sprint drift tactics; Adjunct search using UUVs; conditional, cumulative. and instantaneous probability distributions; and open, reflecting, or semipermiable area boundaries. The analysis will cover a spectrum of realistic operational scenarios including area, barrier, datum,and speed of advance searches against single or multiple, known or unknown targets.

Search planning will comprise plans for sensor employment, the track plan, a speed plan (for constant speed or sprint drift), a depth plan for the ship, a depth plan for the towed array,and a depth plan for adjunct sensors. These plans can consider the search area geometry, the type of acoustic environment, PIM requirements, communication requirements, and average and extremal treatment of multiple target types.

Search evaluation will determine the detection/counterdetection areas and MOEs for the actual conditions, both current and historical (up to the last two days). A target probability distribution is calculated and displayed. The displays allow for sensitivity analysis to parameters such as target speed and target source level variation from the mean. The possibility of background search monitoring to generate situation assessment alerts will be considered.

Search planning and evaluation will require significant data entry in Phases II and III because of the lack of a bus architecture. This should present little problem due to the time available during the search phase.

The human factors engineering for the display of information resulting from the use of these algorithms has to a large extent been completed. The displays have shown themselves to be intuitive and useful in a number of Navy reviews.

4.4 SYSTEM MANAGER TECHNICAL APPROACH

System Manager (SM) supports the other SOAS subsystems by handling all interactions and coordinating all planning activities among subsystems, by providing a consistent global SOAS database, and by monitoring currently invoked plans. Figure 4-18 shows these functions along with the primary inputs and outputs in the SM functional design.

Inputs to the SM are classified into two groups:

1. Environmental information which describes the current state of the submarine and the external threat environment

2. Planning information including proposed tactical and mission plans, and plans representing the Commander's intent

SM outputs are classified into two main categories:

1. Global SOAS information including the threat environment data which is distributed to all other SOAS subsystems

2. Plan coordination information which handles the proposition, invocation, and maintenance of plans throughout the SOAS system

The SM subsystem software design uses a blackboard architecture consisting of a database (the SM blackboard) and knowledge sources which perform SM functions. The SM blackboard contains most of the Global SOAS Blackboard as well as information local to the SM. All communications among the SM's

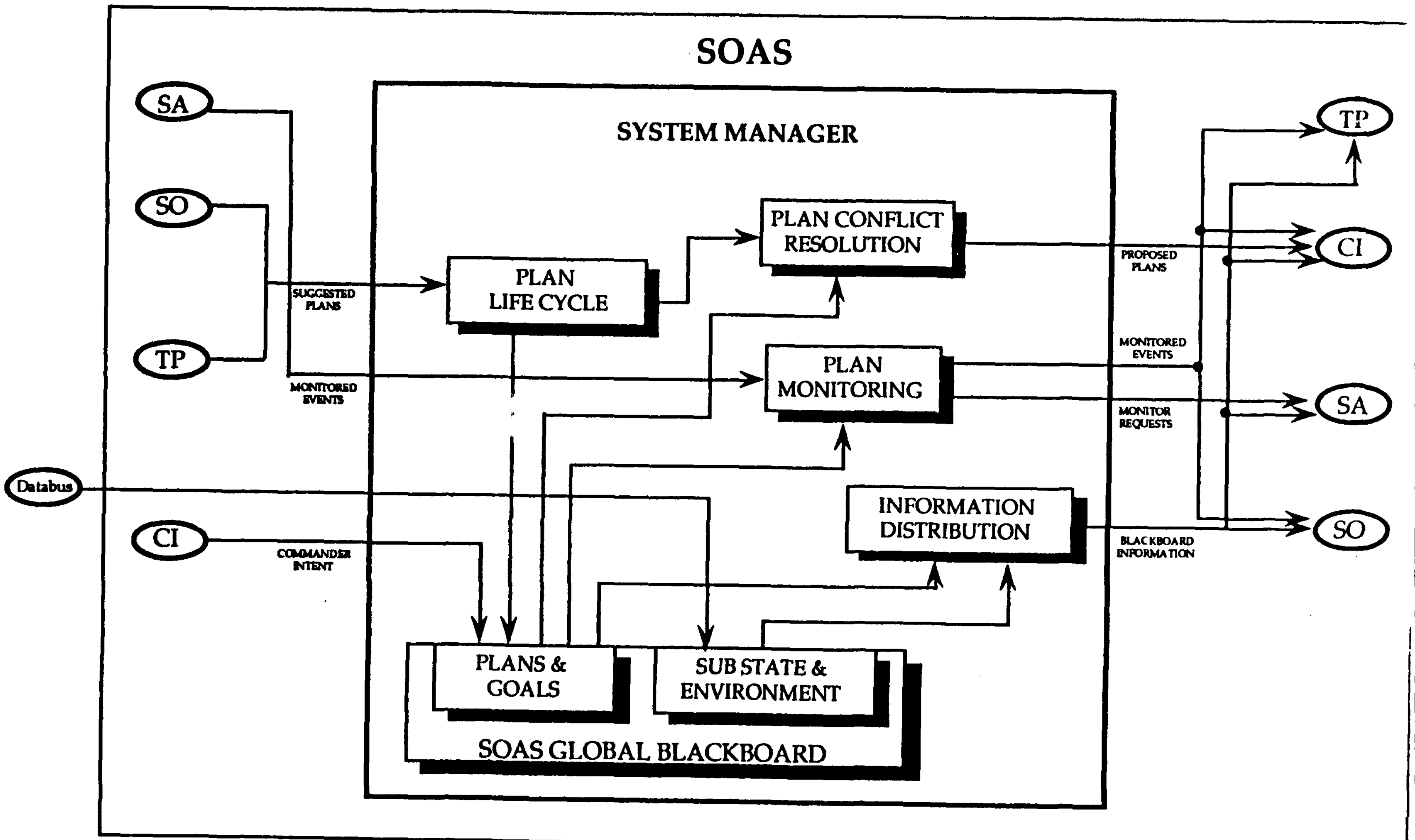


Figure 4-18. SM Primary Functions and Inputs/Outputs

internal modules take place through the SM blackboard. The knowledge sources using the blackboard are segregated into the SM modules described below and shown in Figure 4-19.

All knowledge within the SM falls into one of three domains: interface knowledge, planning knowledge, and control knowledge.

Interface knowledge includes knowledge about how to handle the interfaces with each SOAS subsystem, what information to expect from each subsystem, and how to distribute information throughout the SOAS system. This knowledge is acquired through negotiations with subsystem leads and implementors and through prototype integration exercises. Interface knowledge is documented in the SOAS Interface Control Document (ICD) which defines all messages among SOAS subsystems.

Planning knowledge consists of knowledge for coordinating the progression of plans through the SOAS system, maintaining the plan structures, and monitoring invoked plans. This knowledge exists in the context of the SOAS Plan-Goal Dictionary and the SOAS Plan-Goal Graph. Planning knowledge is acquired through negotiations and through development of the plan progression and approval cycles. Subsystem leads also provide knowledge for the monitoring of plans for which their subsystem is responsible.

Control knowledge includes reasoning about the response time necessary for SOAS subsystem activities and assigning time constraints to subsystem tasks. This knowledge is acquired by considering the SOAS system-level functional timing requirements along with the SOAS subsystem leads' estimates of time requirements for each subsystem task.

The SM software design which has been developed to implement the functions described above can be seen in Figure 4-20. In this design a separate process monitors each subsystem connection for incoming messages. These processes use a queue to pass messages to another process running Inference Corporation's Automated Reasoning Tool (ART). ART contains the rules which implement the SM functions and maintain the SM blackboard.

4.4.1 SM Subsystem Interface Module

The Subsystem Interface module handles the physical interfaces between the SM and each SOAS subsystem. This module's simple I/O utility functions allow it to be implemented entirely in Lisp.

Upon SOAS system initialization, a connection is established over the Ethernet interface with each SOAS subsystem, residing on separate Symbolics computers. For each connection, a dedicated Lisp process is generated which monitors the connection for incoming messages from the other SOAS subsystems. Each message received is stored and the destination address of the message is checked. If the destination address is the SM, then the message is put on the SM input queue which is eventually read by the Blackboard Maintenance module. If the destination address is a subsystem other than the SM, then the message is forwarded directly to the subsystem specified by its destination address.

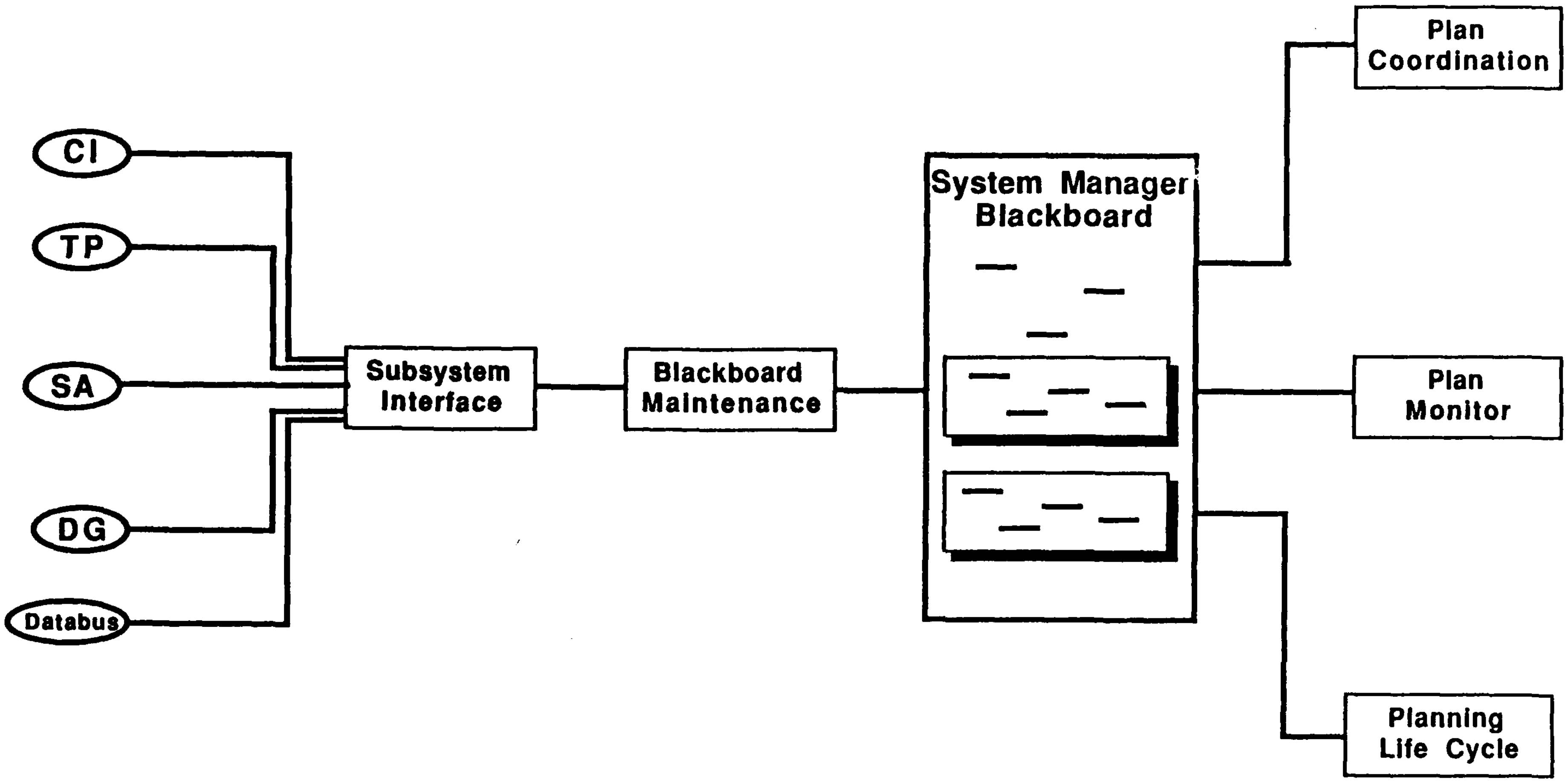


Figure 4-19. <u>SM Functional Design</u>

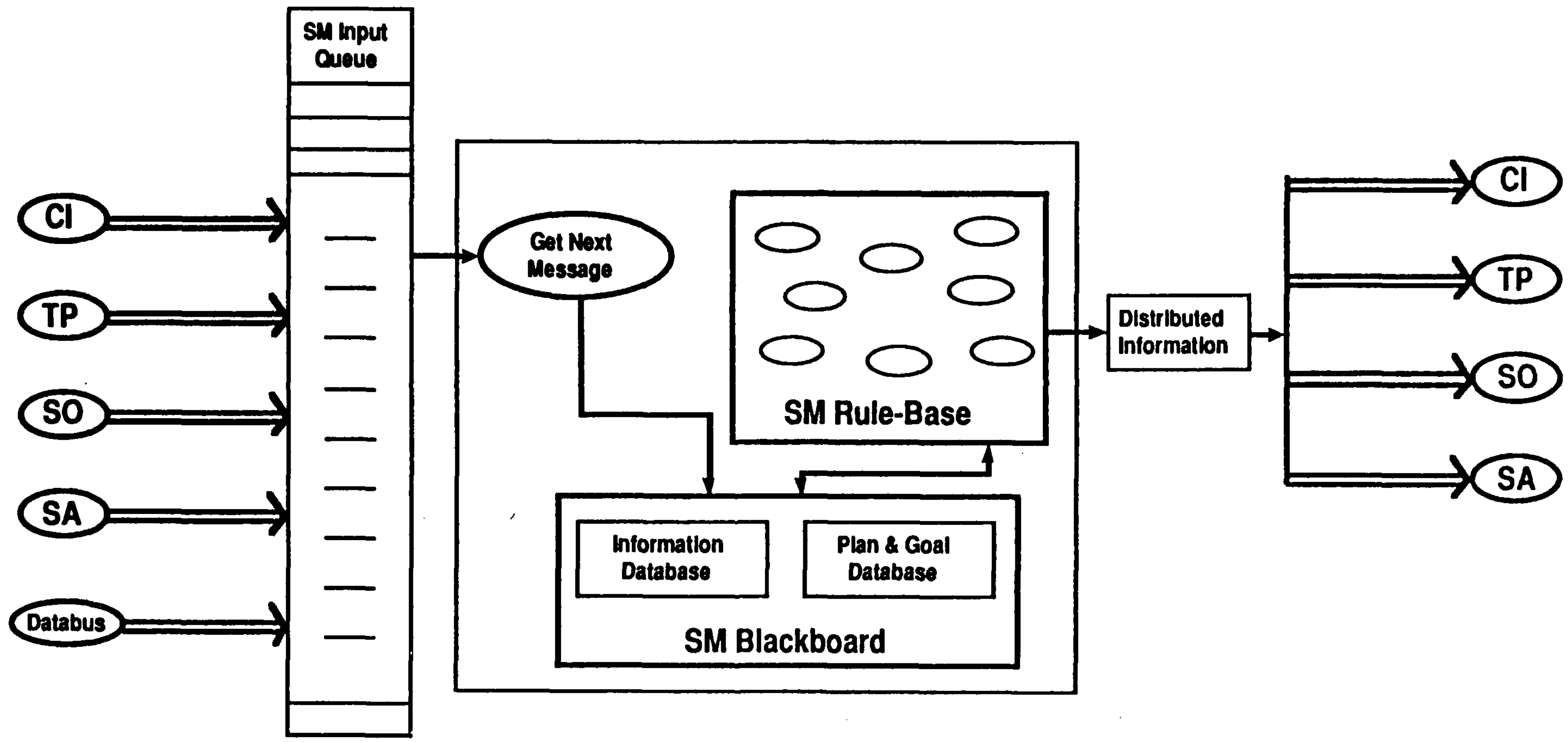


Figure 4-20. SM Software Design

All SM outgoing messages are passed to the Subsystem Interface module and sent out across the subsystem connection specified by the destination address of the message.

4.4.2 SM Blackboard Maintenance Module

The Blackboard Maintenance module maintains the SM Blackboard which contains the Global SOAS Blackboard describing the current state of the world and the SOAS system. Access to information in the blackboard is controlled by this module so as to provide the other subsystems with information both unsolicited and when queried.

The SM Blackboard (Fig. 4-21) contains local facts used internally by SM as well as two databases which comprise the Global SOAS Blackboard:

1. **Information Database** which contains data describing the current state of the submarine, its performance limitations, and its external environment including telecon threats.

2. **Plan Data** which contains both a static and a dynamic plan structure describing plans within the SOAS System

The Plan Database contains static and dynamic plan structures which are tree data structures representing all or portions of the SOAS Plan-Goal Graph. The static plan structure represents the entire SOAS Plan-Goal Graph and contains a-priori information about every plan class. Information stored for each plan class includes its feasibility status, the parent goal which the plan class exists to satisfy, the subgoals established by the plan class, any actions necessary to perform the plan, and monitoring requirements for monitoring an invoked instance of a plan class.

The dynamic plan structure is maintained as plan instances are suggested by SOAS subsystems and invoked by the CI. Each plan instance in this tree points to the specific instance of its parent goal and contains details regarding the progress of the plan instance and the constraints of its subgoals, actions, and monitors.

All data in both the Information and Plan Databases are represented using ART facts. Submarine and environment information exists primarily as "flat' facts. Plan information exists as linked facts forming a tree structure with parameters in one fact pointing to another fact. The tree structure of plans is more suitable to frames (schemas in ART); however, the implementation of the plan trees using "flat" facts out performs an ART schema implementation.

4.4.3 SM Plan Coordination Module

The Plan Coordination module (Fig. 4-22) coordinates the introduction of new plans with currently invoked plans and handles all interactions required during the progression of a plan.

SM Local Facts

- Rule Control Facts
- Target Monitoring Data
- Mission Clock
- Messages

Plan & Goal Database

- Static Plan/Goal Tree
  - Plan Class Names
  - Parent Plan Classes
  - Goals and Subgoals
  - Monitoring Requirements
  - Information Requirements
  - Resource Requirements
- Dynamic Plan/Goal Tree
  - Plan Status
  - Plan & Goal Instance ID
  - Parent Instance ID
  - Monitor Instance ID
  - Information Stream Instance ID

Information Database

- Ownship Performance
- Ownship Position
- Prebriefed Mission Information
- Prebriefed Threat Data
- Battle Area Environment Information
- Mission Progress
- Target Assignment

4-21. <u>System Manager Blackboard</u>

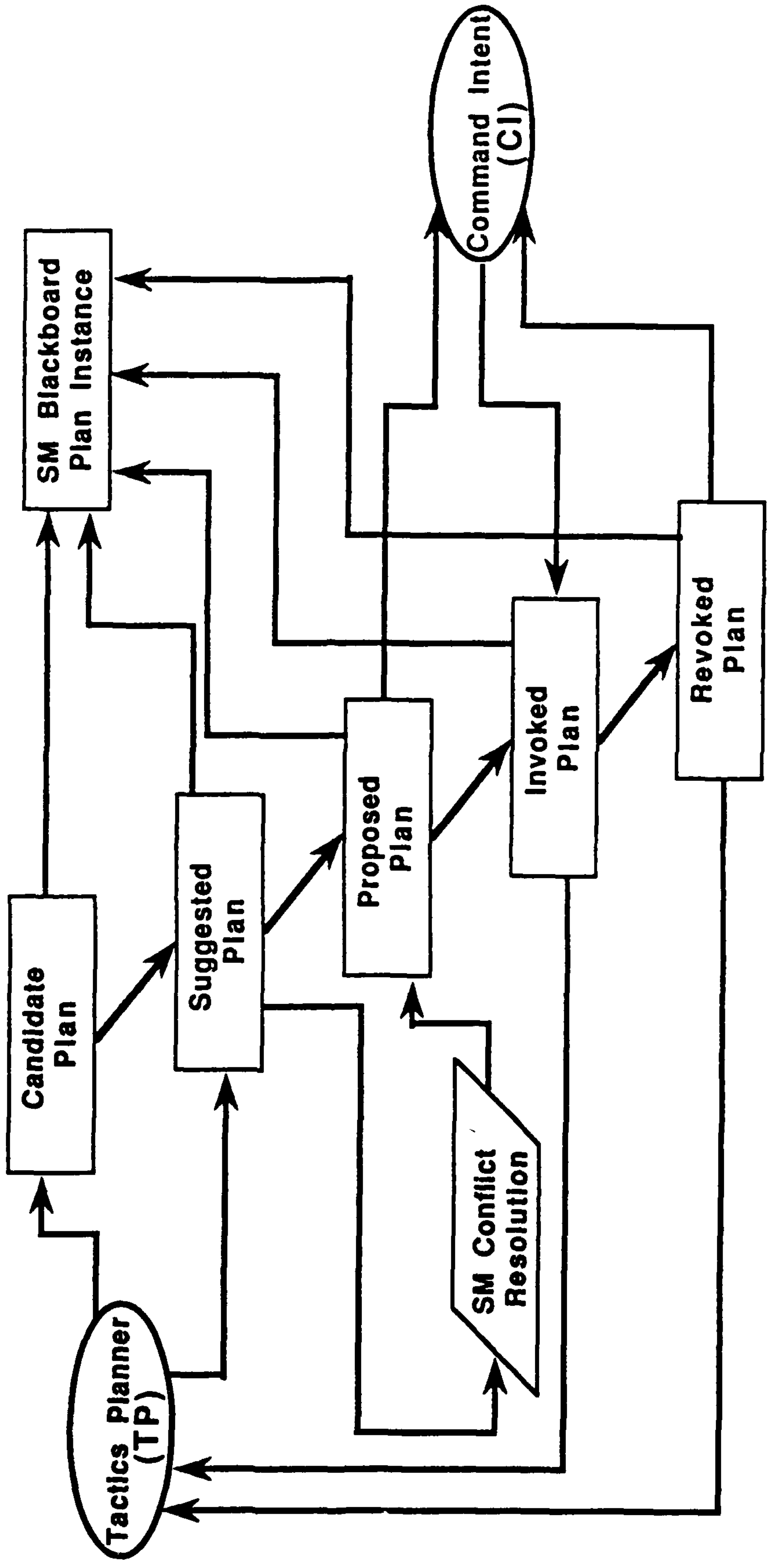


Figure 4-22. Plan Life Cycle

The knowledge needed to perform the Plan Coordination functions is implemented in ART rules. These rules monitor the Plan Database portion of the SM Blackboard as plans progress through the plan life cycle. Once a plan is suggested by a SOAS planning subsystem (TP), the Plan Coordination module looks at the plan's resource requirements to assure that there are no conflicts with the requirements of any currently invoked plans. If free of conflicts, then the plan is proposed to the CI which, in turn, invokes or rejects it.

Knowledge about the resource requirements of a plan and any anticipated conflicting plans is stored as facts with pointers to the plans in the static plan structure to which they apply. This knowledge is acquired through discussions with the SOAS subsystem leads responsible for the generation and use of plans within the SOAS system as well as by observing the interactions between plans during prototype integration exercises.

4.4.4 SM Plan Monitor Module

The Plan Monitor module monitors the constraints under which a plan instance is executing. Achieving a state described by the monitoring constraints of a plan instance signifies that the conditions for completion of this plan instance have been met. Failing to achieve a goal state within some given time or entering a state outside of the limitations specified by the plan instance's constraints signifies the failure of this plan instance and a need to revise the plan or use a different plan.

The completion or failure of a plan instance determines when replanning is necessary. Replanning can be the generation of a new plan due to the completion of a plan, the generation of a new plan due to the unrecoverable failure of a plan, or simply the generation of a revised plan due to failed monitoring constraints. Each of these conditions is determined by the success or failure of constraint monitors associated with a plan instance.

Monitoring and information requirements associated with each plan are stored as part of the static plan structure. As plans become invoked, the SM references these requirements and invokes the appropriate monitors. The SM is responsible for the actual monitoring activities required by some plans; however, the SM uses the SA subsystem to perform the monitoring required by several other plans. For monitoring performed by the SA, the SM sends monitoring and information requests to the SA which help to focus its contact assessment.

4.4.5 SM Planning Control Module

The Planning Control module controls the non-real-time activities of the SOAS System and coordinates these activities to handle the real-time simulation environment.

Future functionality of the Planning Control module will include reasoning about the response time of subsystems in light of the currently active plans, urgency of the situation, and activity of the SOAS subsystems. The SM will be able to assign response-time requirements to each task performed through discussions with subsystem leads to develop a better understanding of the inner workings and processing time requirements of each SOAS subsystem.

5.0 SIMULATION ENVIRONMENT DESCRIPTION

Lockheed developed the Phase 1 Submarine Operational Automation System demonstrations to show the capabilities of the SOAS expert systems during a submarine engagement. These expert systems have the ability to quickly evaluate available system information as well as predicted performance information. Through this evaluation the merit of various tactical alternatives with respect to their risk to Blue are ascertained.

The purpose of this section is to describe the numerous algorithms which were developed and implemented in support of this effort. These algorithms provide a method to simulate the engagement of two submarines. Using current and projected data, they consider the availability and capabilities of submarine systems such that command decisions can be made for any given phase of an engagement.

5.1 OVERVIEW OF THE SCENARIO

This section provides an overview of the Phase I SOAS scenario. It describes briefly the environment chosen for the engagement. It identifies the players and their respective systems, i.e., sensors, weapons, and countermeasures as well as their tactical options. It also provides an overview of the numerous degrees of freedom available in the simulation.

During Phase I, a sample submarine engagement scenario was developed. This scenario formed the basis for identifying requirements, conducting knowledge engineering sessions, and running the SOAS software. The scenario began with two platforms on patrol, Blue and Orange. The Bluet platform had limited a-priori information about the Orange platform in the area from relatively old surveillance data. The Orange platform was controlled by the scenario controller. Since this was an application of expert systems technology and not a mission analysis, the Orange platform had complete knowledge of Blue. Orange knew Blue's current position, course, and speed at all times. The Orange platform's counterdetection ability was calculated and presented to the operator; however, location information was presented prior to counterdetection. Allowing scenario control access to this information permitted the operator to manipulate the scenario to best test the SOAS system by making ad hoc assumptions about the Orange state of awareness.

One of the major factors to consider in the command decision process is how environmental conditions impact sensor performance. Not only does the environment determine the acoustic performance and thus the likelihood of detecting signals, but also the physical attributes of the environment can limit the use of certain sensors. For this scenario, a location in the Sea of O was selected. This choice was based on environmental characteristics and the high probability of future submarine encounters here.

A major characteristic of this environment is the presence of a strong layer that creates a wide variety of tactical options which stress the Commanding Officer's decision-making abilities. The sound generated in a layer is not dissipated but remains within the layer. Therefore, a sensor that is in the layer with the source will have longer-than-usual detection ranges. However, if a sensor is across the layer from the source, then the opposite is true. Detection ranges in the cross-layer case are typically severely reduced. Another major characteristic of this environment is that it is relatively shallow. The maximum bottom depth of this location is 1200 feet, with areas of even less depth in the operating area.

Other major factors to consider in making command decisions are the opposing platform and its respective equipment listing. The key players in this scenario were a Soviet Generic 90's type SSN for Orange and a U.S. Improved 688 class submarine for Blue. Both platforms had low source levels with respect to current platforms. Both were fitted with towed arrays as well as hull-mounted sensors. Blue was fitted with two hull sensors: the BQQ-5C and the Wide Aperture Array (WAA). The Orange platform was equipped with the projected 1990's sensor suite. All platforms were assumed for this demonstration to be restricted to using their hull sensors. The shallowness of the water restricted the use of the towed array; therefore, only hull sensors were utilized. All hull sensors were operated in the passive mode.

Both platforms were also equipped with torpedoes and countermeasures. For this scenario, the blue platform was equipped with regular Adcap, the Orange with an Mk 48 type torpedo. Each platform could fire single or salvo shots. Wire-guided torpedoes were not considered. Blue launched a torpedo based on targeting information from WAA. Orange could launch torpedoes using any information available to scenario control, e.g., line of sight or true course, speed, and position information. Along with launching torpedoes, both platforms used this information for the launching of stationary and/or mobile countermeasures. Countermeasures were generic and were assumed to have a delay onset determined by the launcher. Stand-off weapons were not a factor in this scenario.

5.2 DESCRIPTION OF THE SIMULATION/EMULATION METHODOLOGY

This section describes the simulation of the scenario which was played. It discusses each of the five processes executed to model the submarine engagement: the kinematic control process, the platform controller processes, the detection process, the tracking/TMA process, and the torpedo targeting process. For each of these processes, the algorithm on which it is based as well as input and modeling assumptions are provided. The current input values are unclassified and reflect plausible performance estimates for generic platforms and systems.

The implementation of the simulation was through multiple independent software processes. These processes were executed following the setting of event flags. When an event flag for a process was set, the identified process was then executed and all other processes that manipulate data in common were set into hibernation. While executing, this process had access to all required global data such as truth tables, sensor data tables, and system solution tables. From this information, the process performs the necessary calculations. Once the process has completed execution, the global data is released and the next process in the queue is executed. Figure 5-1 identifies each of the processes executed during this simulation.

Process timer was the controlling process. It maintained the clock used throughout the simulation and defined the order of execution of the five process types identified. These processes are:

- the kinematic control process
- the platform controller processes
- the detection process
- the tracking/TMA process
- the torpedo targeting process

As expected, the kinematic control process determined the kinematic state of each platform. For each platform, it determined the current position in the x, y, and z coordinate system. For each platform, it also determined the current acceleration rate, the course, and the speed. This process was executed for each time step of the simulation.

The platform controller processes defined the orders that were sent for the kinematic process. It consisted of three subprocesses: the torpedo control process, the countermeasure control process, and the Orange control process (SOAS controlled the Blue platform).

- The torpedo control process defined the behavior of the torpedo at pre-enable, search and homing. For each of these torpedo phases, it defined the ordered speed, depth and course of the torpedo.

- Similar to this process was the countermeasure control process. It however, defined the behavior of countermeasures.

- The third subprocess identified was the Orange control process. It allowed the Orange commander to input orders for the Orange platform from a terminal keyboard.

The detection process determined if a sensor detected a platform based on sensor capabilities and environmental conditions. This process was based on the standard figure of merit equation. Short-term variation in signal excess was included in each time step. The detection data generated from this process was then accessed by the next process identified, the Tracking/TMA process.

The Tracking/TMA process predicted a target's course, speed, range, and bearing. This process allowed for lost contacts but did not consider false contacts. (Future phases will add false contacts as well as other required enhancements.) It utilized detection data as well as predefined criteria for track to determine a solution on the target. Targeting was performed on command, rather than on an a-periodic basis.

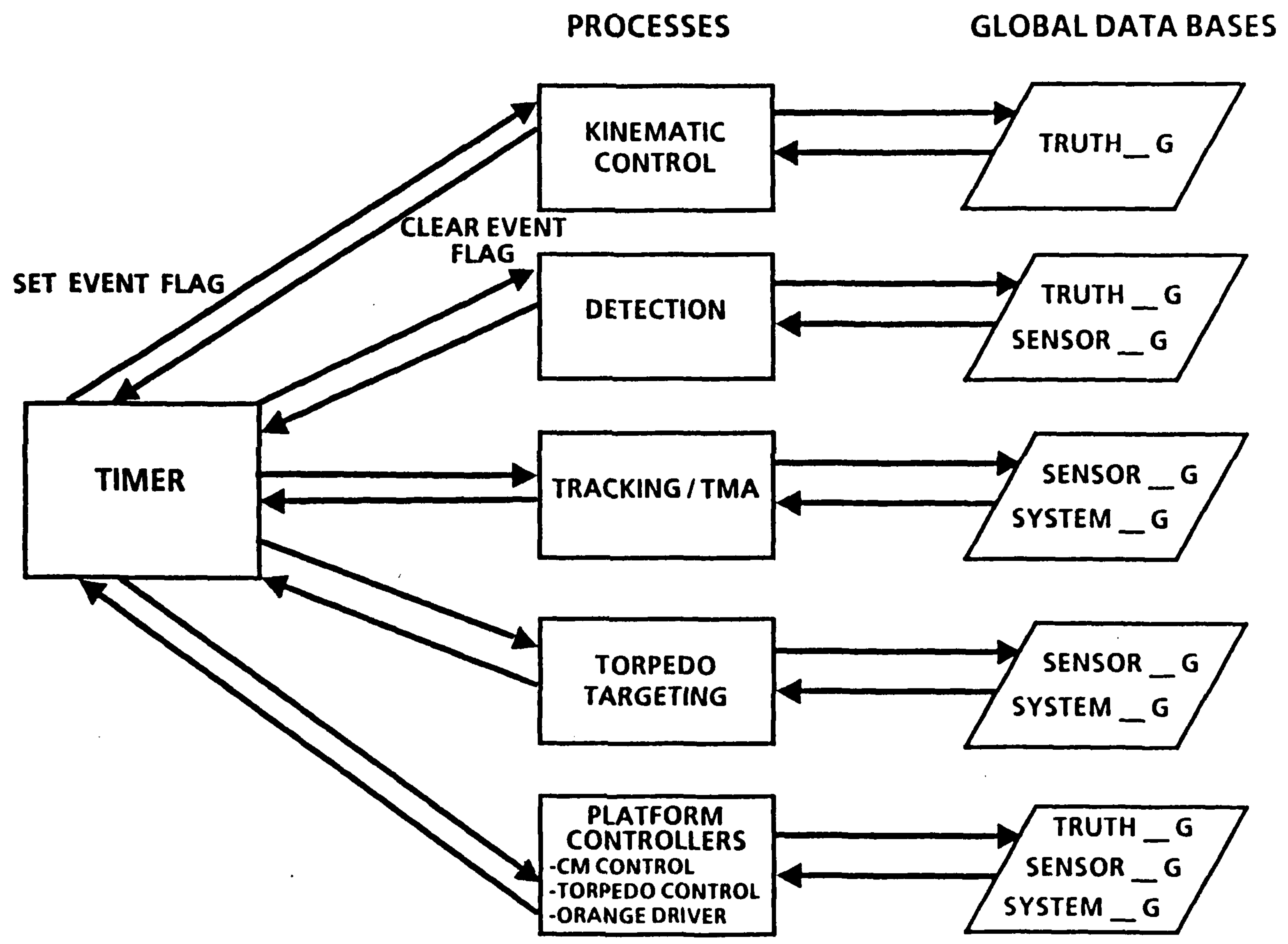


Figure 5-1. <u>Process Control in the SOAS Simulation</u>

The last process identified for this simulation is the torpedo targeting process. Using the predicted target course, speed, range, and bearing, this process determined the appropriate lead angle for weapons launch. It was based on the assumed speed selection presets of the torpedo, the torpedo acquisition capability, and the contact system solution.

5.2.1 The Kinematic Control Process

The kinematic control process determined the kinematic state of all the objects in the scenario at each time step of the simulation. Based on ordered course, speed, and depth, it determined the current position in the x, y, and z coordinate system, the current acceleration rate, the course, and the speed.

Figure 5-2 shows the flow process diagram for this process. Each object was modeled to have the capability to turn, dive, and/or accelerate. The actual degree to which an object moves is defined according to performance specifications. For modeling purposes it was assumed that a platform will change course prior to changing depth and/or speed. Following the completion of a course change, the platform will then change depth prior to a speed change. Following a depth change, a platform will then change speed. Torpedoes change course and speed almost instantaneously, whereas a submarine's maneuverability is more limited.

A submarine's turning radius is a function of a platform's length, rudder setting, and speed. The rate of depth change of a submarine is a function of the dive pitch rate or vertical dive rate and the dive/climb angle. From these parameters, the submarine dive geometry is obtained (Fig. 5-3). Similar geometry is used to simulate a submarine in a climb. For this simulation, the maximum operating depth was assumed to be 1200 feet; the vertical turn rate was 0.5 deg/s/kt, and the dive angle was 25 degrees.

A platform's speed changes are a function of platform type and thrust. Increases in a platform's speed were derived from the following equation:

$$V(t) = VM * (1 - ((T2-t)/(T2-T3))^\wedge AN)$$

where

T2, T3, and AN are constants

VM is the platform's maximum speed.

Decreases in speed were modeled linearly and were at the tactically projected stopping time. Deceleration was a function of initial speed and the time required to come to all stop.

5.2.2 Platform Controller Processes

This process consisted of three subprocesses: the torpedo control process, the countermeasure control process, and the Orange control process. All were responsible for communicating the commanded speed, depth, and course orders for the respective platforms.

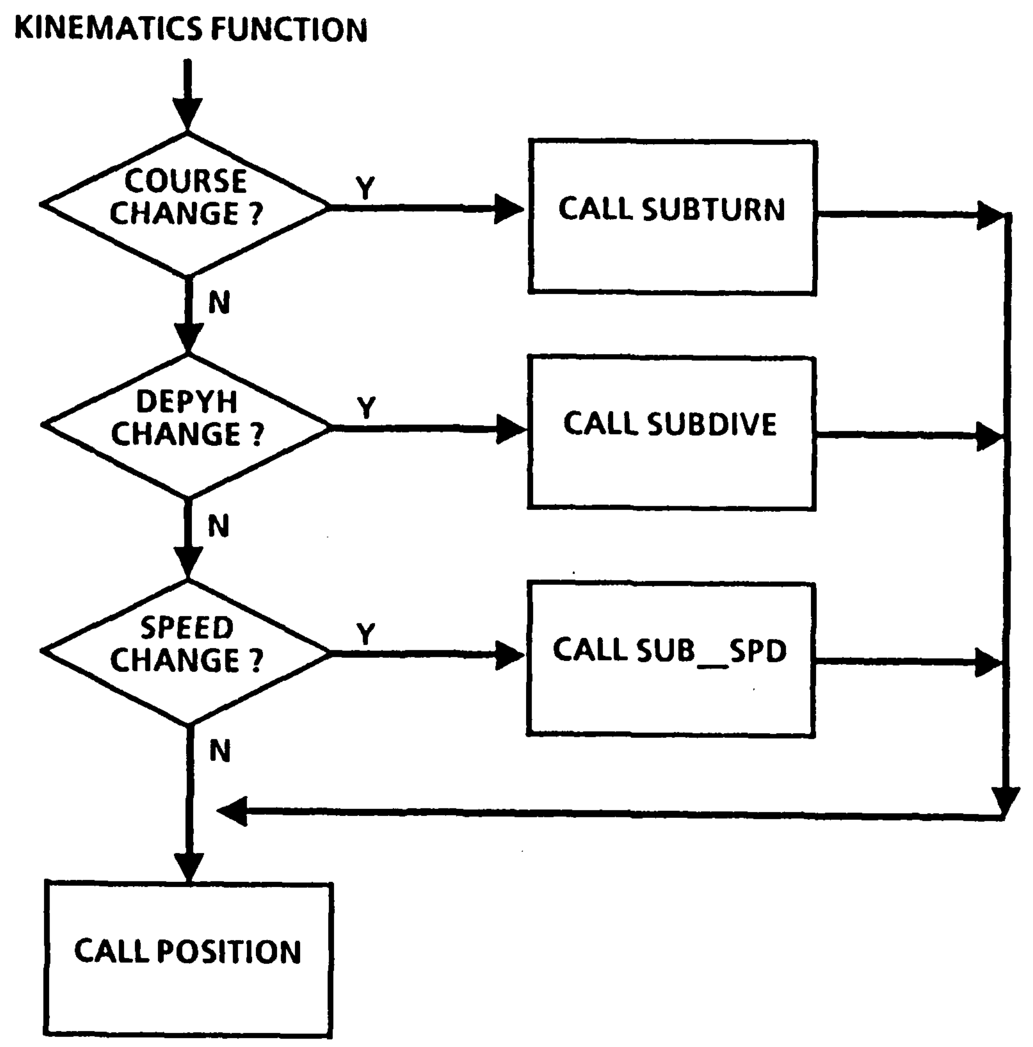


Figure 5-2. Flow Diagram for the Kinematic Process

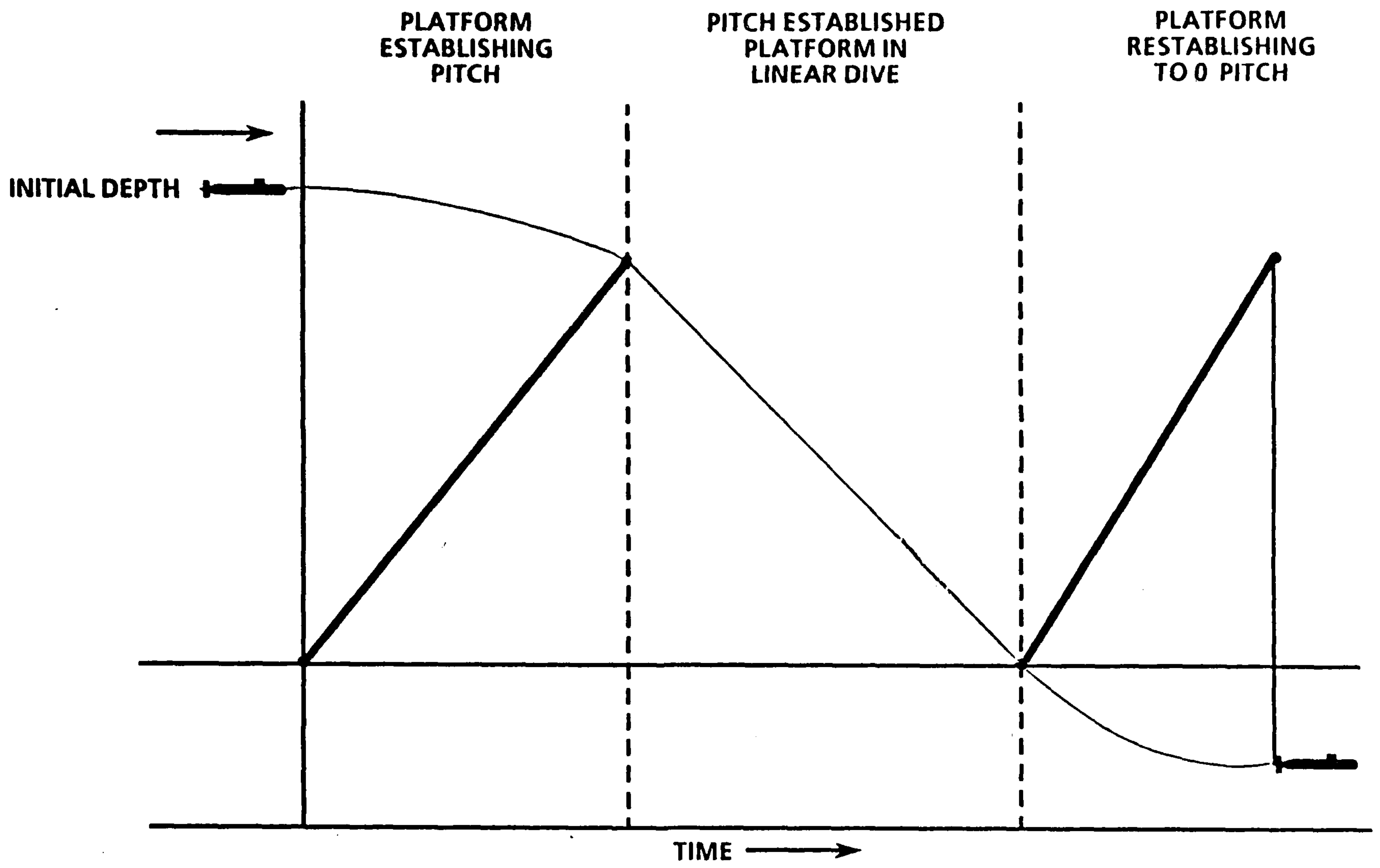

PLATFORM ESTABLISHING PITCH
PITCH ESTABLISHED PLATFORM IN LINEAR DIVE
PLATFORM RESTABLISHING TO 0 PITCH
INITIAL DEPTH
TIME

Figure 5-3. Submarine Dive Geometry

Torpedo Control Process

The torpedo control process determined the behavior of a torpedo up to detonation. It defined torpedo runout, search, and homing based on torpedo presets and the launched angle.

At the time of fire, a torpedo is initialized as a player in the scenario with the same location as the torpedo launcher. The torpedo's speed is set to runout speed; the current course is set to that of the launcher, and the ordered course is set to the inputted gyro angle.

Once the torpedo achieves the ordered course, it continues along this course until the enable point is reached. At the enable point, the torpedo slows to search speed and torpedo sensors are activated. If one or more targets or countermeasures are in the beam, a random draw appropriate to the signal excess is made for each target. When the threshold value is exceeded, the torpedo locks on to the target it is drawing against. This allows for the random selection of a target for closure when more then one target is within the beam at the enable point and allows for the torpedo being drawn off by the other contacts while in homing as the beam width is reduced during homing.

Homing is modeled as a simple pursuit curve. This is accomplished by setting the torpedo's ordered course to the target's bearing at each time interval.

If the target is anything other than a countermeasure, the program checks at each time step to see if the target is within a fusing range. If so, the torpedo explodes and both the torpedo and the target are no longer active.

If the target is a countermeasure and the torpedo overruns it, or if the target is not a countermeasure and the torpedo overruns it without passing within fusing range, then the preset overrun is applied. On overrun, the torpedo circles the overrun target with a predefined radius at search speed and continues to look for a target. If the overrun target moves out of the circle, the torpedo will be re-acquired and attempt to close.

The torpedo continues in this manner until the fuel supply is exhausted which is a function of time and speed.

Countermeasure Control Process

Countermeasures were deposited at the location of the launching platform. The countermeasure could be either stationary or mobile, depending on presets, and was quiet at launch. The countermeasure remained quiet for a preset amount of time; it then began to broadcast. Prior to broadcast a countermeasure had no effect on the scenario. Once broadcasting, the countermeasure was capable of attracting torpedoes in search but was unable to detonate a torpedo. The probability of a countermeasure attracting a torpedo was equally as likely as any other contact, given sufficient signal excess.

Orange Control Process

The Orange operator was provided with a visual representation of the true courses, speeds, and positions of all the contacts in the scenario as well as a VAX terminal running an application program for driving all the players (players were taken to mean any torpedoes, CMs, or platforms) except for the Blue SSN. From this console, the Orange operator had the capability to launch weapons and countermeasures, order course, speed and depth changes, and various other minor functions.

Orange counterdetection capability was provided, but it was up to the Orange operator to act on this information. Since the Orange operator was presented with true positional data, it was up to the operator to emulate the sensors and TMA abilities of the Orange platform. The operator was given the true intercept geometry and was responsible for adding noise to the gyro angle to reflect imperfect information.

5.2.3 Detection Process

The detection process determined the signal excess associated with a hull sensor in the passive mode This signal excess was the difference in dB between the received signal-to-noise ratio and the recognition differential. The signal excess value was a function of the target's radiated noise level as a function of speed, the environment, the background noise level against the sonar system, the directivity index of the array, and the recognition differential. The combination of these terms were defined by the passive sonar equation given below:

$$\text{Signal Excess} = Ls - Nw(r) - Le + (AN - DI) - RD$$

where

+ is for power summation

Ls is the signal level of the target
Nw is the propagation loss as a function of range
AN is the ambient noise
DI is the directivity index
Le is the self-noise
RD is the recognition differential.

A Lamdna-Sigma jump model was used for providing the long- and short-term variations inherent in the system.

5.2.4 The Tracking/TMA Process

This module emulated the processing of information from the sensor up to the system solution table. The module replaced all of the following systems and functions:

- Rapid Ranging Sensor
- Contact Data Correlation
- Target Motion Analysis (TMA)

First, the signal excess information was taken from the detection process. If the signal was greater than zero for a specified time period, then a bearing to the target became available, and a bearings-only contact report and system solution were generated. If the contact was within range, and the geometry of the encounter had an acceptable bearing, then a rapid-ranging solution was generated. This solution, consisting only of the target's bearing and range, along with the time of the event, were entered into a stack in the sensor data table.

This solution was generated by modifying true range and bearing with an error contribution. The magnitude of both the range and bearing errors was dependent upon the signal excess, the relative bearing, and the true range. The greater the signal excess, the closer the relative bearing to 90 degrees or 270 degrees, and the lower the true range, then the smaller the errors for range and bearing. The sensor data table (SDT) was updated at each time step when the signal excess was greater then zero. When there were four or more such records in the SDT, the process began emulation of the TMA function.

TMA was emulated by finding a least squares fit to the datums in the SDT to provide a target course and speed. The results of the TMA emulation were posted in the System Solution Table (SST). It should be noted that several items were repeated in the SST that were present in the SDT (i.e., bearing, range, bearing rate, range rate). When they appeared in the SST, these parameters were temporal data computed from the system solution. When they appeared in the SDT, they reflected the near instantaneous data found by considering the most recent sensor information.

When a contact was no longer held (i.e., Signal excess (SE) was less than zero), the System Solution Table was updated by dead reckoning along the solution path. This provided for a solution on a contact that was coming in and out of sensor range. When the contact had not had a positive SE for a specified period of time, the contact was assumed to be lost and a message was sent to SOAS.

5.2.5 Torpedo Targeting Process

This process determined the appropriate angle for weapons launch using predicted information about the target's course, speed, and range. The angle determined was a function of torpedo presets. It considered the torpedo's minimum pre-enable run, the fraction of the run at laminar intercept at pre-enable speed, the runout, search and homing speeds, and the endurance of the weapon. Figure 5-4 illustrates the geometry for this process.

For each lead angle request, the model determined the time for laminar intercept. Using the law of cosine and the geometry described in Figure 5-4, the time to laminar intercept, TL, was defined to be the positive root, if one existed, of the following equation:

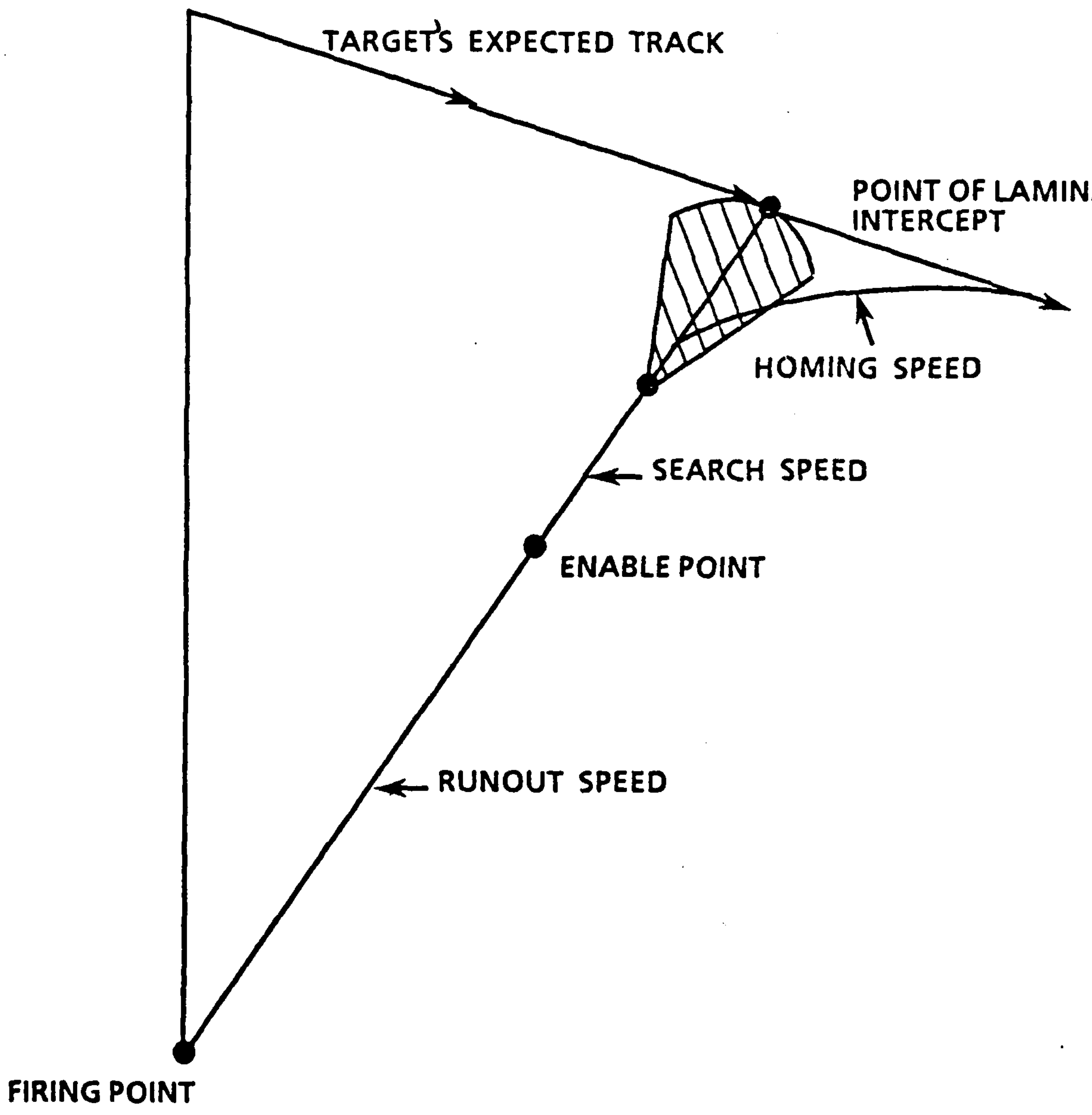


Figure 5-4. Torpedo Intercept Geometry

$$(VO^2-VT^2)*TL+2*(VO*L+FR*VT*COS(AOB))*TL + (L^2-FR^2) = 0.0$$

where

VO	is the torpedo's pre-enable speed
L	is the distance from the target at time of laminar intercept at speed VO
FR	is the range to the target at time of fire
AOB	is the angle on the bow
VT	is the target's speed.

If a positive root was determined, the lead angle, LA, was determined. The equation for determining the lead angle was defined as follows:

$$LA = \arcsin ((\sin(AOB)*VO*TL)/(VO*TL+L)).$$

If no solution was found, the process returned a lead angle equal to the system solution bearing (e.g., line-of-sight solution).

6.0 RELATIONSHIP OF PHASE I PRODUCT TO OVERALL SOAS CONCEPT

The goals that were set forth for this concept development phase were to develop an automation concept to assist the command-level decision makers. The requirements of the Submarine Operational Automation System program are to develop a system that will enhance the mission effectiveness of the attack submarine, to provide demonstrations and evaluations of the benefits of intelligent systems technologies, and to possibly influence the development of technologies by providing an application for them,thus generating a technology "pull" to drive their development.

6.1 SYSTEM OPERATION

The operational relationship between the Phase I product and the the final system is that they are similar in architecture but have differences with respect to the hardware to be used, the language for implementation, and the actual interfaces.

6.1.1 System Architecture

The architecture used during Phase I was based on a distributed blackboard on multiple Symbolic workstations. The Phase II and III architecture will retain the distributed blackboard techniques; however, the system will be located on a single machine.

6.1.2 Predicted SOAS Architecture

As the design of the SOAS moves toward the deployment implementation, the task structure is expected to change. A special-purpose overall scheduler will provide computing resources for processes to be scheduled. This scheduler will allocate resources depending upon predetermined modes in which the SOAS will operate. Individual processes will detect state changes which will cause the scheduler to change modes. The individual scheduler for the subsystems provide permission for their processes to run depending upon the amount of computing resources allocated and the current mode of the system.

Once a process has permission to run, it will actually run when another process detects the event which should trigger the first process. Implicit in this architecture is a hidden constraint on the system design; the sequential nature of the activation of many of these processes must be taken into account in the detailed analysis of the architecture and processing requirements.

6.1.3 Situation Assessment

SA requires the finest-grained scheduling. We assume we are processing multiple possible tracks per second with 100 active monitors. The processing level management will ensure that half of the monitors are used, 65% of the algorithms run and 5% of the estimates for the computing requirements of each, the best estimate for SA over the one hundred twenty seconds is a level 20 MIP process. During the Torpedo Evasion maneuver where the detailed track estimate of the in-bound torpedo must be continuously fed to the Tactics Planner, an additional 7.5 MIPS is required on top of the normal SA functions which must still proceed.

6.1.4 Tactics Planning

The processing load for TP can be divided into three parallel paths. The first manages low-level, spasmodic computing related to management of ship evolution, search planning and attack planning. The second is the constant level of computation associated with the management of the planning process and resolutions of conflicts. The last segment is reserved for high-demand activities such as torpedo evasion.

6.1.5 Command Interface

Three relatively separable paths make up the computing load for the CI. The data manipulation associated with plan management and symbolic messages consumes one path. The more dynamic processing for constitute a second. The third path is fully occupied with the high-load processes associated with information management on the screens.

6.1.6 Hardware

Reasoning functions in real-time computer processing present a major challenge to SOAS hardware and software designers. Therefore, the Lockheed team's approach has targeted the SOAS development toward parallel processing architectures and high-performance VLSI components. Symbolics workstations and IRIS graphics workstations were used in Phase I to facilitate the transfer of the the similar PA architecture into the SOAS environment for concept demonstration. The PA program, just like the SOAS program, recognizes that the use of these workstations are inadequate for a program with large amounts of functionality that runs in real time.

The SOAS choice for the next phases are Solbourne computers - SUN clones that run in parallel. The Solbourne 4/804 is a fully binary-compatible with the Sun-4; in fact it runs the latter's SunOS operating system. Nonetheless, the Solbourne Series4/804 doesn't stop with the complete emulation. This system adds one significant feature not available from any Sun system: multiprocessing. The Series4/804 has four Cypress processors, which function asymmetrical - that is, with one CPU handling the distribution of tasks and I/O. The primary CPU distributes the workload among the CPUs and handles any kernel-level tasks such as file activity and process control or communications. Solbourne includes a fully supported X Windows and X Windows debugger.

6.1.7 <u>Language</u>

The ease of LISP in quickly prototyping AI solutions has been very useful in the PA and SOAS programs. However it is recognized that LISP has a maintenance and performance problem and production problems, so that LISP is not suitable for real time operational use. For these reasons it is necessary to abandon LISP.

Candidate replacement languages considered the most seriously were C++ and Ada. The C++ language offers many of the attributes that are required for the next phases of SOAS without some of the risks involved in going directly to Ada. With regard to Ada, based on a realistic assessment, it is naive to expect that all the risks and problems associated with the marriage of Ada and real-time artificial intelligence are going to be resolved in the next few years. Therefore, the Lockheed team will take an incremental approach using C++.

Progress has begun in the translation of some of the SOAS subsystems to C++. The Pilot's Associate program has similar requirements to go to a real-time system language. The SOAS SA subsystem is over 50% translated. The KADET tool without the SOAS enhancements will be translated by the end of June 89.

6.1.8 <u>Interfaces</u>

The method for communications between SOAS subsystems was by Ethernet message traffic, requiring each subsystem to have a handler for I/O. This mode of operation will be down scoped by having a lot of the data Available through global data files. The passing of message traffic will be incorporated by a DMA read/write operation to reserved message buffer areas for each subsystem.

The communication for Phase I was via Ethernet to a simulation which had data that was processed for the ease of the SOAS subsystems to parse. This will no longer be the case when communication with the HP 9020 Desk Top Computer is interfaced with SOAS to send the CCS Resident Regional Database. The Ethernet protocol will be the same; however, the translation of the information will require more effort.

6.2 <u>ASSESSMENT OF PHASE I PROGRESS</u>

The SOAS working demonstrations showed the potential effectiveness and technical feasibility of a future knowledge-based command and control system for advanced submarines to enhance the tactical and operational performance of the platform. The capabilities that were successfully demonstrated are described below within the context of the original contractual goals established for the program:

> The capability to perform or assist in useful and meaningful submarine command-level functions

- o This feature was shown in the Tactics Planner for a TMA leg while SA was calculating the Measures of Effectiveness for the engagement.

The incorporation and demonstration of innovative and state-of-the-art computer systems, software, and displays

- o This task was accomplished in all subsystem demonstrations and included the use of Symbolic processors, the Lisp language, the KADET planning tool, and the OPAL intent model.

The development, incorporation, and adaptation of innovative and state-of-the-art expert- and/or knowledge-based shells, software, and high-level languages to the submarine command system situation

- o This objective was met through the use of the KADET skeletal planning tool for a dynamic environment, the OPAL intent tool for inferring the intent of not just the command but the individuals implementing the command intent.

Effective man-machine displays and interfaces and the use and incorporation thereof

- o This capability was shown in the CI subsystem demonstration through its assistance assisting in display management and through the actual displays which showed a deeper understanding of the information needed versus the showing of data for crew interpretation.

The compilation and development of a suitable knowledge base of submarine information

- o This requirement was fulfilled in using the databases and models for measures of effectiveness for engagement planning, showing the probabilities for kill, counterkill, and counterdetection for the time now, two minutes in the future, and at the end of the evolution

A system configuration suitable for expansion, revision, and upgrade and interface to ship systems and possibly other expert- or knowledge-based systems

- o This configuration was provided in the SOAS structure, specifically in the planning and assessment methodology. The Plan and Goal Graph and the corresponding Plan/Goal Dictionary are techniques which were first used on the Pilot's Associate program and have since been successfully validated during the SOAS Phase I effort. The proven flexibility of the LASC architecture ensures its continued success in future phases of the SOAS program.